HAL open science

Queue-Management Architecture for Delay Tolerant Networking

Sotirios-Angelos Lenas, Stylianos Dimitriou, Fani Tsapeli, Vassilis Tsaoussidis

To cite this version:

Sotirios-Angelos Lenas, Stylianos Dimitriou, Fani Tsapeli, Vassilis Tsaoussidis. Queue-Management Architecture for Delay Tolerant Networking. 9th Wired/Wireless Internet Communications (WWIC), Jun 2011, Vilanova i la Geltrú, Spain. pp.470-482, 10.1007/978-3-642-21560-5_39 . hal-01583640

HAL Id: hal-01583640

https://inria.hal.science/hal-01583640

Submitted on 7 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Queue-Management Architecture for Delay Tolerant Networking

Sotirios-Angelos Lenas, Stylianos Dimitriou, Fani Tsapeli, Vassilis Tsaoussidis

Space Internetworking Center (SPICE),
ECE Department, Democritus University of Thrace, Xanthi, Greece
{slenas, sdimitr, ttsapeli, vtsaousi}@ee.duth.gr

Abstract. During the last years, the interest in Delay/Disruption Tolerant Networks has been significantly increased, mainly because DTN covers a vast spectrum of applications, such as deep-space, satellite, sensor and vehicular networks. Even though the Bundle Protocol seems to be the prevalent candidate architecture for delay-tolerant applications, some practical issues hinder its wide deployment. One of the functionalities that require further research and implementation is DTN queue management. Indeed, queue management in DTN networks is a complex issue: loss of connectivity or extended delays, render occasionally meaningless any pre-scheduled priority for packet forwarding. Our Queue-management approach integrates connectivity status into buffering and forwarding policy, eliminating the possibility of stored data to expire and promoting applications that show potential to run smoothly. Therefore, our approach does not rely solely on marked priorities but rather on active networking conditions. We present our model analytically and compare it with standard solutions. We then develop an evaluation tool by extending ns-2 modules and, based on selective scenarios primarily from Space Communications, we demonstrate the suitability of our model for use in low-connectivity/high-delay environments.

Keywords: DTN, Queue management, Scheduling

1 Introduction

Queue management in traditional networks is used mainly to regulate traffic fluctuations, as well as to assign priorities to specific traffic classes. It often utilizes dropping mechanisms that signal end-users, implicitly or explicitly, for impeding congestion events. Nevertheless, queue management in Delay/Disruptive Tolerant Networks (DTN) [1] [2], with long delays and disruptions has to address additional issues. While fair resource allocation and traffic classification are still important, queue management needs also to exploit every available contact opportunity and reschedule traffic prioritization and data storage to handle communication disruptions and delays.

We extend further the preliminary architecture proposed in [3] with mathematical analysis, architectural enhancements and systematic evaluation. Our architecture is composed now by three main components: i) an *Admission Control* unit, which determines the criteria that DTN nodes use to accept or reject incoming bundles, ii) a

Buffer and Storage Management unit, which determines how accepted bundles should be stored, and iii) a *Scheduling* unit, which determines bundle service priorities. Here, we refer to data handled by DTN nodes as bundles, even though the architecture proposed is not confined by the standardized Bundle Protocol [4]. The characteristics of each type of DTN network may vary and the objective each time may be different. Here, we emphasize on space applications: space satisfies both dimensions of DTN, that is, disruptions and long delays. We have primarily two goals: i) to increase the DTN device throughput via efficient link exploitation and ii) to increase application satisfaction.

From an engineer's viewpoint, DTN requires additional supportive functionality primarily for resource management. In DTN, resource management incorporates storage capacity as well, which in turn is associated with long delays for data forwarding and increased complexity for scheduling. For example, unlike typical IP network packets, bundles do not face the danger to be dropped, however they do face the danger to expire. Also, priority-marked packets need not prioritized service in case connectivity disruptions have damaged their scope already.

A traditional FIFO-Droptail queue policy may have been an initial candidate for such system. Apart from its simplicity and the fact that it may perform decently in low-traffic networks, this approach is flawed severely: we cannot assign different priorities to different traffic classes and, on top of that, queuing delays punish uniformly and cumulatively all users, even those that may have a chance to survive a potential short disruption. Alternatively, we could integrate a prioritization algorithm in our scheme, such as Priority Queuing (PQ), Fair Queuing (FQ), Class-based Weighted Fair Queuing [5] (CBWFQ) and Low Latency Queuing (LLQ) [6]. A common, undesirable characteristic, however, is that priorities are typically predetermined and/or static, in the sense that do not incorporate connectivity feedback and hence cannot reflect a scheduling policy to a corresponding forwarding implementation. This non-typical requirement renders them only blind tools for DTN management and hence unsuitable, in their present form, for DTN networks. Additional approaches presented in [7] take account the low expiration time left and the number of times a packet has been forwarded, in order to drop it in cases of congestion. The most notable approach is SHLI (drop shortest life time first) which drops the packet with the lowest expiration time. However, this might have some undesirable side-effects when some packets have been delayed significantly, yet we can still manage to forward them before they expire.

Relevant work on DTN queue management is limited, and usually focuses on specific problems, such as policies or scheduling, providing a narrow approach to the queue management, occasionally isolating joint problems. However, some interesting work exists already. In [8], Amir Krifa et al., focus on queue management policies, and reach similar conclusions; they show that traditional buffer management policies such as *drop-tail* or *drop-front* are sub-optimal for use in DTN networks. Current implementations of the DTN, such as ION [9] and DTN2 [10], at this stage, adopt simple approaches to queue management considering the lack of corresponding standards. For example ION, deploys the bundle protocol approach, via a PQ scheme with three queues of outbound bundles, one queue for each of the defined levels of priority ("class of service") supported by BP. Our approach employs an enhanced classification scheme that integrates both network (i.e., connectivity) dynamics and

traffic requirements. Therefore, scheduling is not a product of packet marks and hence, application alone, but rather a joint decision of data priority, application potential to survive disruptions and the network disruptions *per se*.

The rest of the paper is organized as follows. In section 2, we define our queue management model, including only the details necessary for the stochastic analysis, whereas in section 3 we present our stochastic analysis and the corresponding results. In section 4 we apply, validate and compare a part of our model through simulations in ns-2. Finally, in section 5 we conclude and set the framework for future work.

2 Delay Tolerant Queue Management Model

In order to define our queue-management model, hereafter referred as DTQM (Delay-Tolerant Queue Management) we consider the DTN network as a network with low connectivity, high and variable delays and absence of end-to-end path. We define DTQM in a way that allows us to include all the necessary functionality to satisfy a generic set of requirements. Thus, we divide DTQM into three units; i) *Admission Control*, ii) *Buffer and Storage Management*, and iii) *Scheduling*. We discuss all units' functionality, however, due to lack of space we emphasize on the most novel and sophisticated units, namely the *Buffer and Storage Management* and *Scheduling* units.

Admission Control: Admission Control determines how and which data may be accepted from a DTN node and is mainly related to data-custody requests. When custody requests are accepted by a DTN node, that node is obliged to maintain the bundles in its memory, until it is able to forward them, or until they expire.

Buffer and Storage Management: Contrary to traditional networks, where the routing nodes require buffers to implement a store-and-forward strategy, DTN nodes need additional persistent storage to maintain those packets that cannot immediately be forwarded due to limited connectivity. In IP-based routers, the main focus of researchers is to increase channel utilization and decrease delay through scheduling and dropping. This approach inherently assumes that end nodes respond to losses and therefore recover in short time. However, when connectivity is scarce, the requirement for short-time recovery is already violated. Furthermore, DTN networks introduce an additional level of complexity, as a result of combining both volatile and persistent storage. Clearly, the trivial approach to store every incoming bundle in persistent storage and move it to buffer upon request increases the processing delay of all the bundles and fails in cases of applications engaged in low-delay transfers.

In Fig. 1 we depict graphically the Buffer and Storage Management unit. Generally, this model is composed by two units, the Policy unit and the actual Storage unit. The purpose of the Policy unit is to accept all the bundles that enter the node and, depending on the conditions, move them to buffers or storage. Buffer and Storage management is initially differentiated based on whether there is connectivity between the DTN node and the next-hop. During periods of connectivity, packets that enter the node may be immediately routed to the output without being stored first. The total sending rate μ is calculated by the sum of sending rates μ_C and μ_N of the Connectivity and Non-Connectivity buffer, respectively.

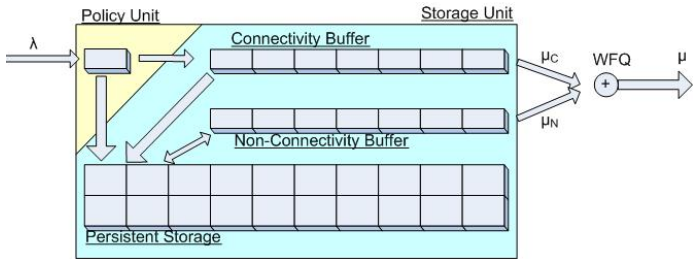

Fig. 1. The Buffer and Storage Management model.

In more detail the purpose of each storage unit can be described as follows:

• Connectivity buffer. The Policy unit moves bundles to the Connectivity buffer only when there is connectivity and therefore the corresponding bundles can be forwarded to the next node. After a time-period which is determined by some threshold, when no connectivity exists, bundles that are stored temporarily in the Connectivity buffer move to Persistent storage.

• Persistent storage. The Policy unit moves bundles to Persistent storage in three cases: i) when there is no connectivity, ii) when there is connectivity but no Connectivity buffer space available, and iii) when there is both connectivity and Connectivity buffer space available, however the contact graph, which is known a priori, instructs that time does not suffice to forward bundles to the next hop.

• Non-Connectivity buffer. Bundles are moved from storage to the Non-Connectivity buffer in the following two cases: i) when bundles are of high priority (are either urgent or a scheduled contact is expected) and there is no connectivity and ii) when there is connectivity but other bundles are selected to be forwarded (opportunistic contact). The algorithm that determines which bundles should be forwarded first at a given communication opportunity is described briefly in the Scheduling section.

Our proposed model additionally deals with the problem of increased processing delay. In the event of multiple nodes in a row that are actively connected, transmission is rather straightforward, since packets are transferred from buffer to buffer without the interference of storage. In the event of short connectivity no further delay due to storage retrieval is imposed; bundles have already moved to the Non-Connectivity buffer, and hence bandwidth through short connectivity can be fully exploited.

Scheduling: Scheduling unit reassigns the priorities for each bundle and determines which bundles should be outputted from the DTN node when a communication opportunity occurs. A priority-oriented model should be inevitably considered; this model should incorporate application requirements, data requirements, Time-to-Live (TtL) for bundles etc.

Although we will not delve into more details, our scheduling depends heavily on the arrival timestamp and on TtL. In order to enhance application service, we promote both packets that have recently arrived in the node and packets that are near their expiration. This approach decreases significantly waiting delays and promotes application satisfaction. In Fig. 2 we depict the priority function used, where ToD

(Type of Data) is a specific identifier that denotes the packet traffic class and TtL denotes the expiration time.

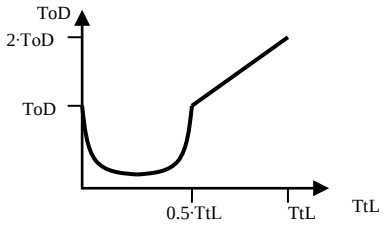

Fig 2. Scheduling priority function.

3 Stochastic Analysis

The purpose of this analysis is to highlight the advantages of our proposed model over traditional scheduling approaches. Our model consists of a primary FIFO queue (Connectivity buffer) and a secondary supportive queue (Non – Connectivity buffer), which serves high-priority bundles. We note that, in the context of Space, the queuing delay involved does not expect to contribute significantly to the total application delay, given the high propagation delay, and furthermore, the potentially very high storage delay involved in typical Space applications. Therefore, a priority queuing (PQ) or a PQ-derived scheduling model for incoming packets does not present conceptually a tempting approach, since it may fail with long-stored packets. Clearly, our approach departs from a FIFO scheme, and therefore, calls for a straightforward comparison with a typical FIFO scheme. However, for completeness, we extend our stochastic analysis also for PQ, which we consider as a theoretical upper bound (only) when connectivity is always present.

It is apparent, from an engineer's perspective, that our model is designed to handle disconnectivity/disruption issues. As such, it is reasonably expected to perform better in environments with limited connectivity, especially against traditional queuing schemes, which do not include a native mechanism to handle intermittent connectivity. Nevertheless, to achieve fairness towards FIFO and PQ, we use a worst-case scenario for DTQM, where connectivity is always available on the system. The results shall indicate to which extent our model is able to perform satisfactorily. In such environments, where connectivity is always available and we do not exploit DTQM's full potential, even a performance comparable to PQ and better than FIFO is acceptable.

We initiate our analysis by modeling network traffic. Packet arrival is modeled as as an exponential process and packet departure as a general distribution process. This is not, however, a globally valid assumption, since different types of traffic can result in different distributions concerning the packet arrival and departure. Nonetheless, as our knowledge on the possible DTN applications is limited, we assume that the environment and the applications under investigation manifest all the necessary characteristics of a M/G/1 system based analysis. The potential existence of self-similar characteristics is not considered here. Furthermore, we assume that all packets

have the same size, non-preemptive priority is enforced and flows correspond to different traffic classes.

One might argue, from an analytical perspective, that precision is rather dubious when we attempt to analytically compare different queuing policies with potentially distinct goals. However, there are several occasions when these mechanisms are indeed equivalent and directly comparable. For example, as throughput is decreasing, the behavior of these three systems converges.

We begin our queuing analysis by estimating the average system delay for each flow in a PQ scheme. We consider a single-server PQ system fed by three Poisson streams with arrival rates λ_1, λ_2 and λ_3. Each stream can be considered as a flow of data generated by various applications, in our case Real-time (RT), Telemetry (TM) and Telecommand (TC) applications. The buffers corresponding to different flows are infinite and packets in each buffer are served in the order they arrive. Thus, we use three queues, one per flow, and three priority classes. We limit the overall system utilization by setting $\rho_i<1$ and $\rho_1+\rho_2+\rho_3<1$. This keeps the system from being overloaded and cancels the possibility of flow starvation. Table 1, presents the notation used throughout the present mathematical analysis.

TABLE I. NOTATION

$\mathbf{N_i}$	Average number of packets in each queue
$\boldsymbol{\lambda_i}$, $\boldsymbol{\lambda}$	Packet arrival rate at class-i queue / Total packet arrival rate
$\boldsymbol{\mu_i}$, $\boldsymbol{\mu}$	Packet service rate at class-i queue / Total packet service rate
$\boldsymbol{\rho_i}$, $\boldsymbol{\rho}$	System utilization factor per class-i / System utilization factor
R	Mean residual service time
$\mathbf{W_i}$	Average queuing delay of a class-i packet
$\mathbf{T_i}$	Average system delay of a class-i packet
$\mathbf{X^2}$	Second service moment

That said, we consider the i_{th} data packet arrival at the first queue of the PQ system. Since class-1 packets have the highest priority, the i_{th} packet that has just arrived must wait in queue for a mean residual time R until the end of the current packet transmission, plus the transmission time required for a mean number of packets N_1 currently in the first queue, preceding the i_{th} packet.

$$W_1 = R + \frac{N_1}{\mu} \tag{1}$$

We calculate the mean residual time, for M/G/1 systems, by the formula ([11]):

$$R = \frac{1}{2}\sum_{i=1}^{n} \lambda_i X_i^2 \,,\ \ X_i^2 = \frac{2}{\mu^2} \tag{2}$$

Now, according to Little's law [12], the average number of packets waiting in the system is equal to the average delay multiplied by the average arrival rate of the system. We apply Little's law to the class-1 queue. As the average queuing delay for class-1 packets is W_1 and the average queue occupancy is N_1 with arrival rate λ_1, we have

$$N_1 = \lambda_1 W_1 \tag{3}$$

$$(1),(2),(3) \Rightarrow W_1 = \frac{R}{1-\rho_1} \,,\ \ \rho_1 = \frac{\lambda_1}{\mu} \tag{4}$$

Similarly, by enforcing non-preemptive priority queuing and according to [11] the average queuing delay for class2 and class-3 packets is accordingly:

$$W_2 = \frac{R}{(1-\rho_1)(1-\rho_1-\rho_2)} \tag{5}$$

$$W_3 = \frac{R}{(1-\rho_1-\rho_2)(1-\rho_1-\rho_2-\rho_3)} \tag{6}$$

Finally, by adding the service time 1/μ of the i_{th} packet in the equations (4), (5) and (6), we can calculate the average system delay for class-k packets:

$$T_k = W_k + \frac{1}{\mu} \tag{7}$$

Next, we estimate the average delay for each flow in a FIFO scheme. In this scheduling scheme, each flow has the same average queuing delay, which depends on the average number of packets in queue N, and the mean residual time, R. According to [11] the average system delay in a FIFO scheme is:

$$T_{FIFO} = W + \frac{1}{\mu} \Rightarrow T_{FIFO} = \frac{1}{\mu(1-\rho)} \tag{8}$$

However, in order to guarantee that the service distribution of the FIFO queue corresponds to an appropriately weighted sum of the service distributions for the different classes in the priority queue scheme, we set:

$$T_{NFIFO} = 3T_{FIFO} - 2R - \frac{2}{\mu} \tag{9}$$

We continue our analysis by estimating the average delay for each flow in a DTQM scheme. We divide the analysis in two parts. Since DTQM uses two outgoing queues (plus the permanent storage – see Fig. 1) one for connectivity and the other for non-connectivity data, we can safely assume that the first one emulates a FIFO queue while the second one approaches the behavior of a PQ scheme. The latter assumption holds since the prioritization function that we apply (see Fig. 2), requires packet sorting. That said, the first queue analysis in terms of average system delay is directly comparable with a FIFO scheme, while the second queue calls for a PQ-based analysis. In order to fairly evaluate our proposed scheme, we omit permanent storage from the mathematical analysis. This allows the three systems to present similar properties and hence be comparable. We consider an average system arrival rate λ equal to the other systems and set arrival rates, without loss of generality, for the two queues $\lambda_a = 0.4\lambda$ and $\lambda_b = 0.6\lambda$. The selection of coefficients 0.4 and 0.6 as the preferred values for our stochastic analysis was based on some initial empirical calculations and will be calibrated further according to emulation results. Finally, the total average system service rate will be μ where $\mu_a=0.4\mu$ and $\mu_b=0.6\mu$ for first and second queues, respectively.

To start off, the average queuing delay for the first queue does not differentiate per flow and can be calculated based on the average number of packets in queue-1 and the

mean residual time, using equation (2) and replacing λ_1, λ_2, μ_1 and μ_2 with λ_a, λ_b, μ_a and μ_b respectively.

$$W_{CON} = \frac{N}{\mu_a} + R \tag{10}$$

By applying Little's law for queue-1 we get:

$$N = \lambda_a W_{CON} \tag{11}$$

From equations (10) and (11) we get:

$$W_{CON} = \frac{R}{1-\rho_a}, \text{ where } \rho_a = \frac{\lambda_a}{\mu_a} \Rightarrow \rho_a = \frac{0.4\lambda}{\mu_a} \tag{12}$$

The average system delay for all the flows in queue-1 is:

$$T_{CON} = W_{CON} + \frac{1}{\mu_a} \tag{13}$$

We now approximate the behavior of the second queue as follows. The queue can split into three sub-queues or subclasses. Each class will be served using a PQ-based scheme. Furthermore, the proportion of packets for each class on the total available capacity of the queue-2 buffer is λ_1/λ for class-1 packets, λ_2/λ for class-2 packets and λ_3/λ for class-3 packets. Therefore, the average queuing delay for class-1 packets is:

$$W_{1_{NONCON}} = \frac{N_1}{\mu_b} + R \tag{14}$$

By applying Little's law [12], we obtain:

$$N_1 = \frac{\lambda_1}{\lambda}\lambda_b W_{1_{NONCON}} \Rightarrow N_1 = 0.6\lambda_1 W_{1_{NONCON}} \tag{15}$$

From equations (14) and (15) we get:

$$W_{1_{NONCON}} = \frac{R}{1-\frac{\lambda_1\rho_b}{\lambda}}, \text{ where } \rho_b = \frac{\lambda_b}{\mu_b} \Leftrightarrow \rho_b = \frac{0.6\lambda}{\mu_b} \tag{16}$$

The average queuing delay for the second subclass depends on N_1 packets, which are buffered in the first subclass, plus N_2 packets, which are buffered in the second subclass, plus the residual time R. In our case, unlike the ordinary properties of PQ, our design assumptions do not permit the possibility of higher priority packets to rearrange the queue at any given stage. Therefore, we have:

$$W_{2_{NONCON}} = R + \frac{N_1}{\mu_b} + \frac{N_2}{\mu_b} \Rightarrow W_{2_{NONCON}} = W_{1_{NONCON}} + \frac{N_2}{\mu_b} \tag{17}$$

$$N_2 = \frac{\lambda_2}{\lambda}\lambda_b W_{2_{NONCON}} \Rightarrow N_2 = 0.6\lambda_2 W_{2_{NONCON}} \tag{18}$$

From equations (17) and (18) we obtain:

$$W_{2_{NONCON}} = \frac{W_{1_{NONCON}}}{1 - \frac{\lambda_2 \rho_b}{\lambda}} \tag{19}$$

By the same token, average queuing delay for third subclass is:

$$W_{3_{NONCON}} = \frac{W_{2_{NONCON}}}{1 - \frac{\lambda_3 \rho_b}{\lambda}} \tag{20}$$

Finally, system's average delay for each subclass is:

$$T_{k_{NONCON}} = W_{k_{NONCON}} + \frac{1}{\mu_b} \tag{21}$$

Having calculated the average delays for the two queues separately, we will combine these values to acquire the total delay. A statistically acceptable method for doing this is by using weights. Considering the way that we have defined the problem, it is logical to expect that each queue will contribute with a different percentage to the overall system delay. Hence, we will consider that queue-1 and queue-2 will contribute to the total average system delay by 40% and 60%, respectively. Considering the values of λ and μ in each queue, we have.

$$T_{1_{DTQM}} = 0.4T_{CON} + 0.6T_{1_{NONCON}}\ ,\ T_{2_{DTQM}} = 0.4T_{CON} + 0.6T_{2_{NONCON}}\ ,\ T_{3_{DTQM}} = 0.4T_{CON} + 0.6T_{3_{NONCON}} \tag{22}$$

As for the numerical results presented below, the value of system service rate is constant at 10 packets/sec, whereas the values of λ vary in order to obtain the possible range of system utilization, 10% - 90%. The value of service rate, considering packet sizes in the order of KB, provides an acceptable rate of data transmission, especially in space environments and among low energy sensors.

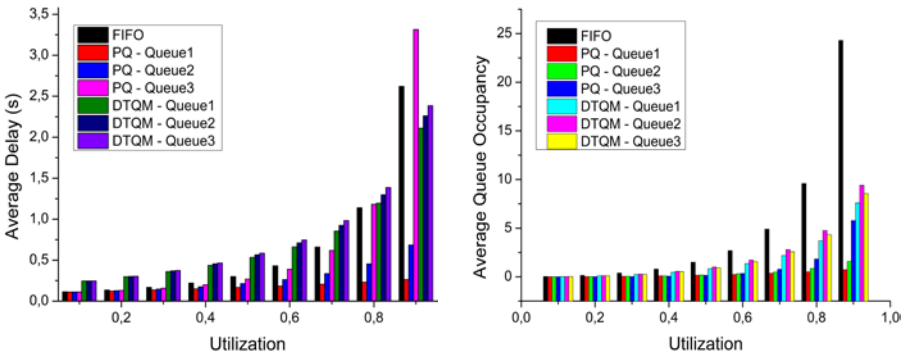

Fig 3. Numerical results

The numerical results of our analysis are presented in Fig. 3. We compare the aforementioned queuing schemes based on the average delay for each queue in each

system and the average queue occupancy. The results show that our approach achieves a performance clearly better than FIFO and in some cases better than PQ, especially when the system utilization factor is high.

4 Experimental Evaluation

Evaluating such a queue management policy requires extensive experiments using an actual space network, since DTQM affects both lower-level (battery lifespan) and higher-level (throughput, latency) performance metrics. We implement and evaluate our proposed solution using the ns-2 software simulator [13]. This implementation was not trivial and required significant time considering the fact that ns-2 does not support DTN. In this work, we focus on the full implementation of the second and third component of DTQM, namely the *Buffer and Storage Management* and *Scheduling* parts, leaving the rest of the architecture evaluation as future work.

We apply DTQM to the network topology of Fig. 4. We assume three sending nodes, S_1, S_2, S_3, which are located in space and send traffic generated by various applications (Real-time, Telemetry and Telecommand; all constant bit rate with 0.02 sec, 0.4 sec and 60 sec sending interval respectively) and a receiving node R, which is located on earth. All traffic generated by the sending nodes is routed through the routing node Q in which we deploy DTQM.

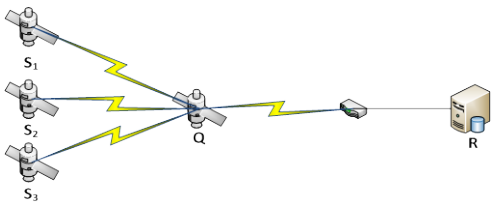

Fig 4. Simulation topology

Since we assumed that the sending nodes are located in space, the connectivity of the wireless links should be intermittent. In order to emulate a DTN environment where connectivity is not predetermined, thanks to alternative routes and connections, we select a random disconnectivity pattern with uniformly distributed connectivity disruptions spread across the entire duration of the experiment, as the most appropriate for evaluating the proposed architecture. In this context, link S_1-Q is uniformly unavailable in total 1% of the time of the experiment. Similarly link S_2-Q is 5% unavailable, link S_3-Q is 10% unavailable, and finally, link Q-R is 0.5% unavailable. Furthermore, the propagation delay of links S_1-Q, S_2-Q, S_3-Q and Q-R was set respectively to 2sec, 600 sec, 300 sec and 0.6 sec. Finally, we set the total time of the experiments to one hour, and measure the total number of received packets for all three sending nodes, as well as the *Application Satisfaction Index* (ASI) [14] of the network, a metric that highlights the contribution of the queuing delay to the total delay. In order to add reliability to our results and enforce randomness to take effect we repeat the experiment several times. In particular, the performance of DTQM was evaluated in five connectivity scenarios, each one utilizing a different (randomly generated) connectivity schedule. We compare the obtained results with the corresponding results of a FIFO policy. In line with the priority function used (see Fig. 2) we assign ToD for

each application as follows: 1 for the RT application, 2 for the TM application and 3 for the TC application.

Furthermore, we set the packet intervals for each application as we would expect in real-life, that is, Real-Time packets are generated with the shortest interval and Telecommand packets are generated with the longest interval; hence the difference in packet numbers.

In Fig. 5 below, we present the results from the comparison of DTQM against Droptail, using *ASI* and *average delay* as our performance metrics. The first observation that we can make is that DTQM outperforms Droptail in any case. In particular, we experience a 60% delay decrease on average and in some cases it can reach up to 90% reduction of the corresponding bundle delay using a typical FIFO scheme. This delay decrease is also reflected on the system ASI, which is increased 20%, on average.

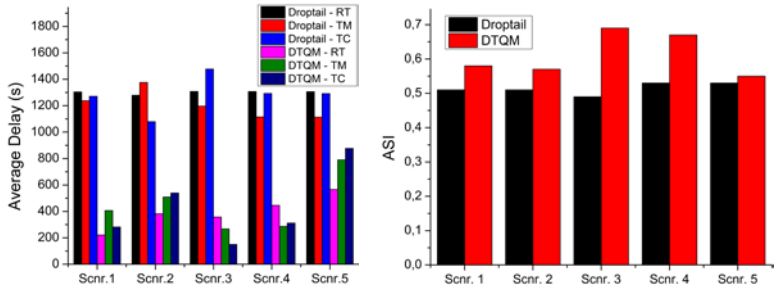

Fig 5. Experimental results

TABLE II. EXPERIMENTAL RESULTS

		Best Case			Worst Case		
		RT	**TM**	**TC**	**RT**	**TM**	**TC**
Received packets	**FIFO**	15680	696	5	15732	645	4
	DTQM	15618	754	5	15500	869	7
Average Delay	**FIFO**	1308	1196	1477	1306	1113	1219
	DTQM	357	267	150	565.9	789.7	877
System ASI	**FIFO**	0.49			0.53		
	DTQM	0.69			0.55		

Table 2 demonstrates in detail the best and worst case results for DTQM. By viewing Table 2, we notice that the received packets are almost the same in any case (with the exception of the TM packets of the worst case experiment) regardless of queuing policy. Nevertheless, we may experience up to 40% increase in received Telemetry and Telecommand packets when DTQM is deployed, since we assign them with higher ToD. Moreover, the most interesting observation is the undoubtable improvement of the average delay regardless of the application. However, since we transfer the same number of packets in both cases, how can we justify the delay decrease? DTQM uses a sophisticated scheduling algorithm that assigns higher priority to the packets most recently arrived in the node and promotes them in the queue (see Fig. 2). Thus, packets are reordered in the buffer based on the time they entered the routing node. Classic scheduling uses a rigid FIFO approach, which although it seems to promote fairness, in fact it increases the communication time, with the risk of dropping a packet due to TtL expiration.

5 Conclusions

In this paper we proposed a novel architecture for queue management in DTN nodes. Although the available space confines us from presenting a more detailed version and evaluation of our model, we sketched several of its characteristics.

One of the most interesting results was initially introduced by the stochastic analysis, which yielded positive results for the operation of our model that exhibits a behavior far superior to FIFO and comparable to PQ, even in network conditions that are unfavorable for our scheme. We also obtained supportive results from the conducted experiments that alleviate worries from adopting numerous assumptions on the stochastic analysis section. Therefore we can safely claim that DTQM has the potential to achieve smaller queuing delays and higher application satisfaction when connectivity is scarce.

Our next step is to enhance our evaluation towards two directions: i) to present a more detailed analysis, which incorporates total capacity and storage capacity as well, in order to highlight one major property of DTN and ii) to extend the experiments by using the space-oriented testbed [15] that we have developed in our lab.

Acknowledgments. *The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013_FP7-REGPOT-2010-1, SP4 Capacities, Coordination and Support Actions) under grant agreement* ***n° 264226*** *(project title: Space Internetworking Center-SPICE)*

References

1. Fall, K.: A delay-tolerant network architecture for challenged internets. ACM SIGCOMM 2003, Karlsruhe, Germany, 25-29 August (2003).
2. Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., Weiss, H.: Delay Tolerant Networking Architecture. RFC 4838, April (2007).
3. Dimitriou, S., Tsaoussidis, V.: Effective Buffer and Storage Management in DTN Nodes. E-DTN 2009, St. Petersburg, Russia, 14 October (2009).
4. Scott, K., Burleigh, S.: Bundle protocol specification. RFC 5050, November (2007).
5. Class-based Weighted Fair Queuing, Cisco IOS Software Releases 12.0 T.
6. Low Latency Queuing, Cisco IOS Software Releases 12.0 T.
7. Lindgren A. and Phanse K. S.: Evaluation of queuing policies and forwarding strategies for routing in intermittently connected networks. *Proc. of IEEE COMSWARE*, January 2006.
8. Krifa, A., Barakat, C., Spyropoulos, T.: Optimal buffer management policies for delay tolerant networks. IEEE SECON 2008, San Francisco, California, June 16-20 (2008).
9. Jet Propulsion Laboratory: ION: Interplanetary Overlay Network. https://ion.ocp.ohiou.edu.
10. The Delay-Tolerant Networking Research Group (DTNRG): http://www.dtnrg.org/.
11. Bertsekas, D. P., Gallager, R.: Data Networks. Prentice Hall (1991).
12. Little, J. D. C.: A Proof of the Queueing Formula L = λ W. Operations Research, 9, (1961).
13. Network Simulator, http://www.isi.edu/nsnam/ns/ (1997).
14. Mamatas, L, Tsaoussidis, V: Differentiating services with Non-Congestive Queuing (NCQ). IEEE Transactions on Computers, 58(5):591–604 (2009).
15. Samaras, C. V., Komnios, I., Diamantopoulos, S., Koutsogiannis, E., Tsaoussidis, V., Papastergiou, G., Peccia, N.: Extending Internet Into Space - ESA DTN Testbed Implementation and Evaluation. Mobilight 2011, Athens, Greece, 18-20 May (2009).