On Temporal Composability of Cyber-Physical Disruption-Tolerant Networks

Fatemeh Saremi and Indranil Gupta

Department of Computer Science
University of Illinois
Urbana, IL 61801
Email: {saremi1, indy}@illinois.edu

Abstract

Disruption-Tolerant Networks (DTNs) enable transfer of data when network nodes are only intermittently connected. With respect to the scarcity of connectivity in DTNs, a key research challenge is to guarantee schedulability of data flows which are to be fed into the network and prevent network resources from being channeled into serving data packets which cannot make deadline. Research efforts in real-time community are all concerned with traditional type of resources which are not available in time, but available in space. However, the nature of DTNs resources deviates from this model: not only are the network resources temporally unavailable, but also spatially unavailable. This distinct paradigm of resources demands for specialized schedulability analysis techniques to be developed. To this end, among a wide range of problems real-time DTNs require to be studied, in this paper we take the first step in this direction and analyze schedulability of the workload in recurrent DTNs. We extend the recently proposed delay composition framework for disruption-tolerant networks and provide a worst-case estimation of end-to-end latency of data flows and investigate how latency is composed in DTNs. We then analyze whether or not the multi-criticality spatially-distributed data flows meet their end-to-end deadline constraints. Our study through simulation based experiments show that our approach has moderate pessimism and provides appropriate schedulability guarantees prior to establishment of the distributed workload.

Keywords: Disruption-Tolerant Network (DTN), Latency, Schedulability.

1. Introduction

Disruption-Tolerant Networks (DTNs) [1] depend on spatio-temporal intermittent connectivity for transmission of data. The intermittent connectivity may be a result of different environmental and/or device-induced factors such as nodes' movement, limited radio range, power management, node failure, malicious attacks, etc. DTNs have a wide range of applications from interplanetary applications to terrestrial applications such as ecological monitoring, meteorological data transfer, disaster recovery, naval networked systems, vehicular networks, rural areas usability, web caching, email/not so instant messenger, etc. Although there is no guarantee of continuous in time end-to-end connectivity under DTN-like paradigm of networking, in space interaction of nodes in this model of networks provides some notion of connectivity and feasibility of data transmission which is performed based on the mechanism of *store-carry-and-forward*.

DTNs' challenges have been a hot research topic in networking community during recent years. A wide variety of routing protocols have been designed for these networks and issues related to mobility models, congestion control mechanisms, energy concepts, transport protocols, etc. have been addressed as well [2]. However, while extensive cyber-physical applications of DTNs (such as naval applications) have been proposed and deployed, investigating timing properties and workload schedulability of these networks have not been scrutinized yet.

The large number of nodes and their significant dynamism and complicated behavior make accurate analysis of the temporal properties too complicated and inefficient. In this paper, with regard to the fact that DTN entities inherently move in a recurrent manner, we consider a disruption-tolerant network with recurrent movements. Each network node recurrently meets other nodes, and when in contact, they transfer data packets to one another. Each node retains data

This work was funded in part by the Assured Cloud Computing Center UCOE grant from AFOSR and AFRL.

packets until it encounters another node to which it can transfer the custody of the packet in order to being eventually delivered to destination (i.e., flow sink). The decision as whether or not to transfer the packet is made by the routing protocol and lies beyond the scope of this paper.

We take an end-to-end view of the DTN and provide an efficient upper bound for end-to-end latency of the distributed data flows. We exploit delay composition framework, a recently developed framework in real-time community for traditional form of distributed resources which are not available in time, but always available in space [3]. However, DTNs violate this model, especially that such a behavior is not an exception in DTNs, but rather their essential working characteristic. There exist challenges in these networks that must be addressed in a schedulability analysis technique designed for this kind of distributed systems, e.g. there is no well defined notion of execution time in this type of distributed systems. We extend the framework for recurrent DTNs in which network resources suffer unavailability in space as well as in time.

We provide an upper bound estimation of the worst-case end-to-end delay and investigate latency composability in recurrent DTNs. We then reduce the entire network to a single virtual node, on which the end-to-end delay of data flows is no less than the original distributed network. Any uniprocessor schedulability analysis test can then be exploited to assess the schedulability of the original workload in the DTN.

The rest of this paper is organized as follows. The previously proposed models emulating DTNs as well as schedulability analysis approaches designed for regular distributed systems are presented in Section 2. The network model and the problem statement are expressed in Section 3. Our analysis of the worst-case end-to-end latency bound of DTN data flows and schedulability analysis approach are elaborated in Section 4. Evaluation methodology and experimental results are provided in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

The timing properties of DTNs such as schedulability and real-time capacity have not been scrutinized before. In this section, we provide an overview of proposed models for DTNs as well as schedulability analysis techniques for regular distributed systems.

2.1. DTN Modeling

[4] models the DTN as a time-dependent graph and uses a modified version of the Dijkstra algorithm to find the minimum delay path in the graph. [5] models the network with a space-time graph consisting of different layers, each of which represents a copy of the DTN in one time slot of the network's activity and denotes encounters in that time slot. Then a shortest path algorithm can be utilized to calculate the path with minimum delay.

[6] considers a deterministic mobility and solves the routing problem subject to network constraints utilizing algorithms based on the model introduced in [4]. [7] [8] concentrate on a non-deterministic DTN with periodic mobility and model the network as a space-time graph. Then they construct a probabilistic state-space graph and obtain the optimal routing scheme in terms of expected delivery latency as a solution of a Markov decision process. [9] considers a deterministic and centralized DTN and constructs a time-independent graph. Then they exploit graph algorithms to achieve optimal results subject to network constraints.

[10] introduces inter-contact graph, a new model for DTNs which captures the notion of phase of encounters. The authors propose a probabilistic routing protocol which exploits more reliable routes and controls message replication thereof. The authors in [11] explore the single-copy routing space and propose a theoretical framework to derive upper and lower bounds on the delay. The authors use this framework to analyse the performance of multi-copy routing scheme as well [12].

2.2. Schedulability Analysis in Distributed Systems

Accurate analysis methods such as [13] [14] construct a precise schedule of length equal to hyper-period and suffer from huge time complexity in large distributed systems thereof. Offline schedulability tests [15] [16] [17] ignore parallelism of executions and divide end-to-end deadline of tasks into per-stage deadlines which results in significant pessimism.

Holistic schedulability analysis [18] [19] [20] considers the worst-case delay of each stage as the jitter of next stage. Network calculus based approaches [21] [22] [23] [24] [25] analyze the distributed network one node at a time. They model arrival pattern of flows at a particular node and the node's scheduling policy using arrival and service curves, respectively. Based on this information, the rate of departing flow which would serve as the arrival curve for the next node can be determined.

Delay composition algebra [26] [27] [28] [3] [29] which is a reduction based approach, benefits from considering execution overlap of tasks running in a distributed system to provide an upper bound for delay of the tasks.

However, the analysis approaches in literature do not perfectly match the paradigm of shared resources in DTNs.

3. Network Model and Problem Statement

Let $S =< N, E, F >$ be a disruption-tolerant network in which N denotes the set of participating nodes, E denotes the set of communication links, and F denotes the set of data flows in the network.

Each node $n_i \in N$ moves in the network and encounters other nodes recurrently. For example, the network could be considered as a post-disaster scenario that rescue workers, volunteers, survivors, etc. perform a mobility of recurrent nature, e.g. rescue workers move periodically between emergency operations centers and survivors to aid them and provide survivors with medical supplies and emergency survival kits.

Each communication link $e_{i,j} \in E$ connecting two nodes n_i and n_j denotes that two nodes meet each other more frequently than a given number of times within a specified time window. Each link $e_{i,j}$ has a label $T_{i,j} \in [T_{i,j}^{min}, T_{i,j}^{max}]$ which is the time interval between (the beginning of) every two encounters between node n_i and node n_j. The concept is depicted in Figure 1. In the figure, nodes n_i and n_j meet each other every $T_{i,j}$ time units, nodes n_i and n_k contact each other every $T_{i,k}$ time units, and nodes n_j and n_k have rare (if any) encounters with each other.

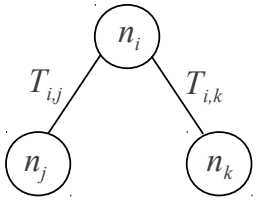

Figure 1. Communication graph

Each real-time data flow $f_i \in F$ has a deterministic route $< n_{i_0}, n_{i_1}, n_{i_2}, ..., n_{i_{l_i}} >$ wherein n_{i_0} and $n_{i_{l_i}}$ are respectively source and destination of the flow, and l_i is the length of the path of flow f_i. The communication path of f_i is shown in Figure 2. Throughout the paper we will use the equivalent notation $< enc_{i_0,i_1}, enc_{i_1,i_2}, ..., enc_{i_{l_i-1},i_{l_i}} >$ to depict the path of flow f_i and denote the set of encounters through which f_i is being transferred, wherein enc_{i_0,i_1} denotes the encounter of node n_{i_0} and node n_{i_1}, and so on.

Figure 2. Path of flow f_i

Each flow is assigned a priority. The scheduling policy can be either static (such as rate-monotonic) or dynamic (such as EDF) and packets belonging to the same flow may have different priorities. However, the priority of a particular packet, when determined, remains unchanged throughout the network.

When two nodes come into communication range of one another, they exchange their bit vectors that indicate the packets they are currently carrying. Then, they will transfer packets in priority order. Therefore, as far as higher priority packets exist to be transferred, lower priority ones cannot be transmitted and will be delayed.

Data flows can be either periodic, sporadic or aperiodic. The union of the path of all flows forms the workflow graph of the network. Each data flow f_i has an end-to-end deadline D_i and the data packets of f_i must be delivered within this timing constraint.

The duration of each encounter and the communication bandwidths are restricted and only a limited number of packets can be transferred in each contact. The maximum number of packets that can be transmitted in encounter $enc_{i,j}$ is equal to $w_{i,j} \in [w_{i,j}^{min}, w_{i,j}^{max}]$ which depends on environmental and network conditions. The $w_{i,j}$ can be simply represented using the encounter duration, link bandwidth, and size of data packets, however, for simplicity we assume that it denotes the maximum number of packets that can be transferred.

There might be more than one packet belonging to the same flow that delay a particular packet of another flow. In addition, more than one packet of a flow might be simultaneously present in the network. The latter happens when the end-to-end deadline of the flow is larger than its period. The goal is to determine if the real-time flow-set is schedulable or not, that is, whether or not data packets belonging to different flows can be delivered within their prespecified end-to-end deadline.

4. Temporal Composability and Schedulability of DTNs

DTNs are different from regular networks as communication links are intermittently formed and thus exploiting resources in these networks is very restricted. In this section, we propose an approach for analyzing schedulability of a set of spatially-distributed multi-criticality flows to ensure feasibility of the network's real-time traffic load.

To determine schedulability of the flow-set, the end-to-end delay of each flow should be calculated to examine if packets from all data flows can meet end-to-end deadline constraints or not. In this section, we elaborate on our method acquiring an effective upper bound for the delay of each flow as well as the schedulability analysis of the real-time flow-set.

The approach reduces the entire DTN into a single virtual node with worst-case end-to-end latency of packets being no less than that of the original DTN. The equivalent workload on the virtual single node can be easily obtained from the characteristics of the workflow. The approach scales well as the network size increases and enjoys moderate pessimism which makes it suitable for real deployments of DTNs.

4.1. DTN Delay Composition

In this section, without loss of generality, we fix some flow f_i in the flow-set and determine an effective upper bound on the worst-case end-to-end latency that a packet p from f_i incurs in a disruption-tolerant network. We take into account the contribution of one packet from any other flow into delay of p. Then, we apply a uniprocessor analysis technique to account for further packets from other flows that may contribute to delay of p. In other words, the uniprocessor schedulability test acts as a separate layer that captures how many packets from other flows contribute to delay of p. The goal in this subsection is to effectively bound the worst-case delay of packet p due to a single packet from any other interfering flow. In the next subsection we shall illustrate how the uniprocessor analysis can be employed.

The packets that can affect delay of p are only the ones whose lifetime overlaps the response time of p, i.e. the time interval starting from the arrival time of p at source node until the delivery time of it at destination node. Even more, the only situation that one packet can delay another one is when both are to be simultaneously (i.e., during the same encounter) transmitted. Therefore, flows which do not share any encounter have no effect on delay of one another. Besides, even the delay because of such flows is solely due to their common encounters with f_i. Thereupon we take into account every consecutive set of common encounters separately and ignore other parts.

Let $F = \{F_H, f_i, F_L\}$ in which F_H denotes the set of flows with higher (or equal priority) than f_i and F_L denotes the set of flows having priorities lower than f_i. Let SF_k be the set of all subflows sf_k^y of f_k where sf_k^y denotes a maximal set of consecutive encounters shared by f_k and f_i. For example, assume that routes of f_i and f_k are $< enc_{4,5}, enc_{5,9}, enc_{9,11}, enc_{11,12}, enc_{12,17}, enc_{17,20}, enc_{20,18}, enc_{18,16} >$ and $< enc_{4,5}, enc_{5,9}, enc_{9,10}, enc_{10,11}, enc_{11,12}, enc_{12,15}, enc_{15,16}, enc_{16,18}, enc_{18,20}, enc_{20,24} >$ respectively. Then, we have $SF_k = \{< enc_{4,5}, enc_{5,9} >, < enc_{11,12} >,$

$< enc_{20,18}, enc_{18,16} >\}$, $sf_k^1 = < enc_{4,5}, enc_{5,9} >$, $sf_k^2 = < enc_{11,12} >$, and $sf_k^3 = < enc_{20,18}, enc_{18,16} >$. Let p_k^y denote the representative packet of subflow sf_k^y.

The interfering subflows can be classified into three categories: *forward subflows*, *backward subflows*, and *cross subflows*. The forward subflows are the ones that traverse the consecutively shared set of encounters in the same order as f_i (such as sf_k^1 in the aforementioned example). The backward subflows are those that traverse the set of common encounters in the opposite order as that of f_i (such as sf_k^3). The cross subflows share only one encounter with f_i (such as sf_k^2). Let SF_F, SF_B, and SF_C denote all (higher or equal priority) forward, backward, and cross subflows respectively.

Let $ict_{x-1,x,x+1}$ be the time interval between two encounters enc_{i_{x-1},i_x} and $enc_{i_x,i_{x+1}}$, called inter-contact time. The encounter graph of the path of f_i is depicted in Figure 3 (corresponding to the path shown in Figure 2). The loop from every encounter to itself indicates that the encounter happens recurrently and the associated label specifies the period/inter-encounter time. The arc between two different encounters, say from enc_{i_{x-1},i_x} to $enc_{i_x,i_{x+1}}$ depicts that node n_{i_x} takes packets of flow f_i from node $n_{i_{x-1}}$ at encounter enc_{i_{x-1},i_x}, and then $ict_{x-1,x,x+1}$ time units later encounters node $n_{i_{x+1}}$ for the first time. At this time, node n_{i_x} can transfer packet p to $n_{i_{x+1}}$ only if less than $w_{i_x,i_{x+1}}$ packets of higher (or equal) priority are present for transmission at that encounter. Otherwise, node n_{i_x} retains the packet until the first future encounter $enc_{i_x,i_{x+1}}$ that the packet can be transferred.

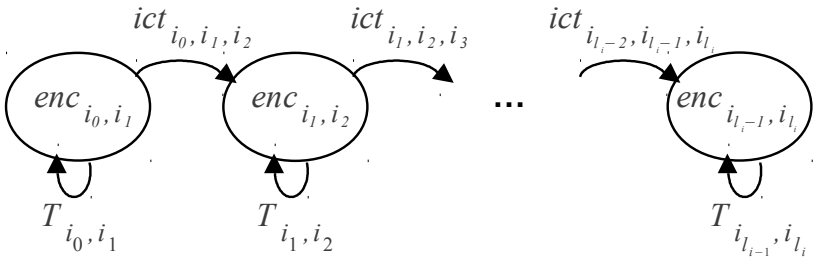

Figure 3. Encounter graph of f_i.

Note that the arcs between two different encounters are not necessarily direct and each node might encounter some other nodes in between two encounters. However, each arc can be seen as direct in the sense that it is connecting two immediate encounters during which the custody of f_i is being transferred, that is, both encountering nodes are custodians of f_i.

As mentioned, when node n_{i_x} cannot transfer packet p at encounter $enc_{i_x,i_{x+1}}$, it must wait for the first future time that the resource (i.e. encounter $enc_{i_x,i_{x+1}}$) is available for p to be transferred. As depicted earlier, the resource $enc_{i_x,i_{x+1}}$ will be available at most every $T_{i_x,i_{x+1}}$ time units. Therefore, the delay imposed on f_i in order to access the resource $enc_{i_x,i_{x+1}}$ will be $T_{i_x,i_{x+1}}$ time units for every $w_{i_x,i_{x+1}}$ higher (or equal) priority packets sharing the resource with p. The concepts are graphically depicted in Figure 4.

The figure shows transmission trace of packet p. At each encounter $enc_{i_x,i_{x+1}}$, the delay due to every $w_{i_x,i_{x+1}}$ packets of higher (or equal) priority packets is represented with a *delay block* of the length equal to $T_{i_x,i_{x+1}}$. The last delay block has length equal to $ict_{x,x+1,x+2}$ time units and denotes transmission of a group of packets along with packet p. The number of packets being transferred through such encounter is not necessarily the maximum permissible (as opposed to previous encounters in the same busy period). The node identifiers depicted at top of the figure indicate the particular custodians of packet p which are carrying it during the corresponding time interval.

One approach to obtain the end-to-end delay is to calculate the worst-case delay due to higher priority packets in each encounter separately and add up those to transmission delay of p. This will report an additive delay which does not effectively take the transmission overlaps into account. However, we prove that the DTN delay is sub-additive and the additive approach is significantly pessimistic. It is because of the fact that a higher priority flow f_k delays f_i not in all shared encounters. Specifically, we prove that it is only due to a single common encounter. DTN Delay Composition Theorem follows.

DTN Delay Composition Theorem. *In a DTN with recurrent mobility and work-conserving scheduling policy in which priority order is immutable across encounters, the worst-case end-to-end delay of a packet of flow f_i can be bounded*

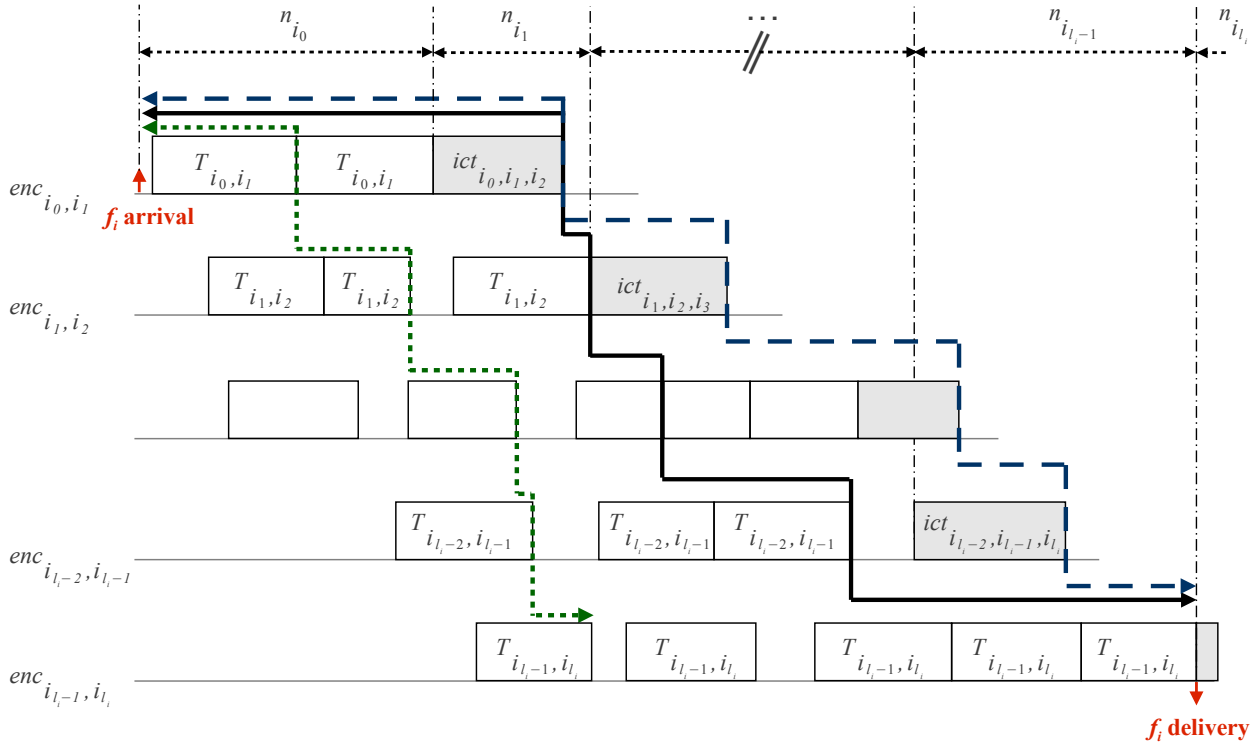

Figure 4. The trace of delay and transmission for packet p among custodians.

as follows:

$$Delay(f_i) \leq \sum_{x:0\leq x\leq l_i-1} T^{max}_{enc_{i_x,i_{x+1}}} + \sum_{k:f_k\in F_H} \sum_{y:sf^y_k\in SF_k} \max_{x:enc_{i_x,i_{x+1}}\in sf^y_k.path} \frac{T^{max}_{enc_{i_x,i_{x+1}}}}{w^{min}_{enc_{i_x,i_{x+1}}}} \tag{1}$$

Proof. Consider the delay trace of a packet p from f_i with following conditions: (i) starts at the arrival time of p at the source node and ends at the delivery time of p at the destination node; (ii) contains a sequence of contiguous intervals, each of which with nonzero length and related to one encounter, starting at the encounter immediately after the preceding interval on the previous encounter ends, i.e. successive intervals belong to consecutive encounters of the path of f_i; (iii) each interval ends after the first delay block on the corresponding encounter which has the following property: at least one of the packets (which might be p itself), say p^z_j, being transferred through that delay block either (a) shares at least one future encounter with p, where the associated delay blocks containing p^z_j and p are connected to one another without any idle time in between (i.e. either side by side or through a set of other delay blocks), or (b) shares at least one "future" encounter with some intermediary packet, say p^y_k, having properties (a) or (b) , where the associated delay blocks containing p^z_j and p^y_k are in a consecutive set of delay blocks in that encounter with no idle time in between. We call a trace with aforementioned properties *first traversal delay trace* or simply first traversal trace. In Figure 4, the first traversal trace is denoted in solid line. The reason for the necessity of condition (iii) in definition is to avoid traces which enter into idle time and terminate before delivery of p (such as the dotted trace in Figure 4).

The existence of a first traversal trace is guaranteed, since a trace that goes to next encounter after the delay block representing transmission of p among other packets is a potential first traversal trace (i.e. the dashed trace in Figure 4). It is evident that length of the first traversal trace correctly bounds the delay of p in the network.

Two types of delay blocks contributing to the trace can be distinguished. First, the delay blocks which correspond to the transmission of first group of packets, at least one packet of which either directly or indirectly (via an intermediary packet through recursion, i.e., condition (iii), property (b)) impacts the delay of p in the future. Second, the delay blocks that do not belong to the first category. The delay contribution of each category to the first traversal trace follows.

The number of delay blocks in the first category is no more than the number of encounters on the path of p. The reason is that any such delay block is the last block at some interval of the first traversal trace that corresponds to one encounter on the path of p.

$$\sum_{x:0\leq x\leq l_i-1} T^{max}_{enc_{i_x,i_{x+1}}} \tag{2}$$

To bound the delay of the second category of blocks, we consider three aforementioned classes of subflows (cross, backward, and forward subflows) and calculate the contribution of each one into the length of this category.

A cross subflow shares only one encounter with f_i and thus delays p in at most one encounter. Hence, the contribution of each cross subflow into the length of second category is at most the common encounter's period/inter-encounter time. In addition, it is worth noticing that all delay blocks of second delay category represent transmission of the maximum allowed number of packets in the particular instance of the corresponding encounter. Therefore the delay due to each subflow is a fraction of the length of the delay block. Hence the contribution of all cross subflows into the length of the second category in the first traversal trace is bounded by:

$$\sum_{x,y:sf^y_k\in SF_C, enc_{i_x,i_{x+1}}\in sf^y_k.path} \frac{T^{max}_{enc_{i_x,i_{x+1}}}}{w^{min}_{enc_{i_x,i_{x+1}}}}. \tag{3}$$

A backward subflow affects the delay of p in only one encounter as well. Without loss of generality, assume that the backward subflow sf^y_k of higher (or equal) priority shares encounters $< enc_{i_{x-1},i_x}, enc_{i_x,i_{x+1}}, enc_{i_{x+1},i_{x+2}} >$ with f_i, and p^y_k is currently interfering p at encounter $enc_{i_x,i_{x+1}}$. After being transferred at $enc_{i_x,i_{x+1}}$, p moves to the next step and waits for the next resource on its path to become available, i.e. $enc_{i_{x+1},i_{x+2}}$. On the other hand, p^y_k moves one step ahead towards accessing encounter enc_{i_{x-1},i_x}, which is the next resource of p^y_k's path. Therefore, after delaying p at encounter $enc_{i_x,i_{x+1}}$, p^y_k moves away from p and no longer delays it. Thus each backward subflow delays p in at most one encounter. With respect to the fact that each delay block of second category denotes transferring the maximum permissible number of packets in the encounter, the contribution of all backward subflows to the length of second delay category in the first traversal trace is hence bounded by:

$$\sum_{y:sf^y_k\in SF_B} \max_{x:enc_{i_x,i_{x+1}}\in sf^y_k.path} \frac{T^{max}_{enc_{i_x,i_{x+1}}}}{w^{min}_{enc_{i_x,i_{x+1}}}}. \tag{4}$$

In the delay expression, the maximum delay contribution over the common encounters is considered because of the fact that the interference of two flows, the backward subflow sf^y_k and flow f_i, may occur at any of the shared encounters.

Now we bound the contribution of forward subflows into the second category. By definition, none of the packets being transmitted through such blocks delay p in the future. It means that any such block is the "last" contributing block (into delay of f_i) for all packets being transferred through that block. Therefore each forward subflow can contribute to at most one of such delay blocks. Therefore, the delay contribution of all forward subflows into the first traversal trace is bounded by:

$$\sum_{y:sf^y_k\in SF_F} \max_{x:enc_{i_x,i_{x+1}}\in sf^y_k.path} \frac{T^{max}_{enc_{i_x,i_{x+1}}}}{w^{min}_{enc_{i_x,i_{x+1}}}}. \tag{5}$$

The reason for the max function in the delay expression is that in the worst-case, the contributor may occur at the encounter with maximum value for the ratio of maximum period/inter-encounter time to minimum bandwidth.

Adding up delay expressions given in equations (2)-(5) completes the proof.

4.2. Schedulability Analysis

Using the DTN delay composition theorem, any of the traditional schedulability analysis methods for a single node (such as the Liu and Layland test [30], the hyperbolic bound [31], and the exact tests [32], [33]) can be utilized to investigate schedulability of the real-time flow-set in the DTN. The last term in the theorem gives the delay due to higher priority subflows present in the system during the lifetime of f_i. The formula proposes to consider a single hypothetical node which models the timing properties of the entire network. The workload on the hypothetical single node is described as follows:

- Each higher (or equal) priority subflow sf_k^y in the DTN is replaced with a (virtual) single-node subflow of the same end-to-end deadline and period as that of flow f_k and the "execution time" equal to:

$$\max_{x:enc_{i_x,i_{x+1}} \in sf_k^y.path} \frac{T^{max}_{enc_{i_x,i_{x+1}}}}{w^{min}_{i_x,i_{x+1}}},$$

- Flow f_i is replaced with a (virtual) single-node flow of the same end-to-end deadline and period as that of flow f_i and the "execution time" equal to:

$$\sum_{x:0 \leq x \leq l_i - 1} T^{max}_{enc_{i_x,i_{x+1}}}.$$

The entire procedure of estimating the worst-case delay bound errs on the safe side. The operation of cutting a flow into subflows, each consisting of a set of consecutive shared encounters, and focusing on such parts does not decrease the end-to-end delay of f_i. Specifically, the delay is increased since an adversary would have more space to choose the arrival time of each subflow and increase the delay. The delay of each flow in the original network is no more than that in the virtual single node and the reduction errs on the safe side as well. Therefore, the result of schedulability test on the single hypothetical node can be applied to the original distributed DTN. It means that if the schedulability test for the hypothetical single node succeeds, the original flow-set in the distributed network is schedulable as well.

5. Evaluation

In this section, we evaluate our approach in terms of two measures of interest: average fraction of data flows admitted and average ratio of end-to-end delay to delay bound. We use our approach as an admission controller and compare the results in presence and absence of the controller. The admission controller admits as many flows as it can deem to be feasible. In the absence of the admission controller, the DTN does efforts to deliver all packets to their destinations within their predetermined deadlines. The admission controller assists to prevent network resources from being channeled into serving packets which may not meet deadline.

We model a request-reply scenario in which a source node sends a request to a destination node through a sequence of intermediary nodes and a reply from the destination follows the same sequence of intermediary nodes. In each experiment sufficient number of flows are fed to the AC to take it into the overload region. The overload region is of interest since it reflects the capability of the approach to deliver flows on time. The default number of nodes in the network is 15. The period of occurrence of encounters are chosen as $10^{1+x} \times MR$, where x is a variable with uniform distribution over the interval $[0, EPR]$ with EPR denoting the encounter period resolution and having a default value of 2, and MR is the mobility resolution, the default value of which chosen equal to 2. This choice of encounter periods enables them to vary by a factor of 10^x. The end-to-end deadline of each flow is chosen as $10^{1+x} \times N \times DR$, where N is the number of encounters on the path of the flow and DR is the deadline resolution, set to 30.

We use deadline monotonic scheduling as the uniprocessor schedulability policy. The results of each experiment are averaged over 50 executions and a total of 50000 requests/replies (from different flows) are generated during each execution. The 95^{th} percentile confidence interval for all values represented is within 5% of the mean value and is not depicted in the figures for the sake of legibility.

Figure 5 evaluates the scalability of our approach and compares the average fraction of data flows admitted by the admission controller (AC) with the average fraction of data flows delivered on time in the absence of the admission

controller (w/o AC). As the figure shows, the pessimism of our approach does not increase when network size increases and this fact makes it beneficial to large networks as well.

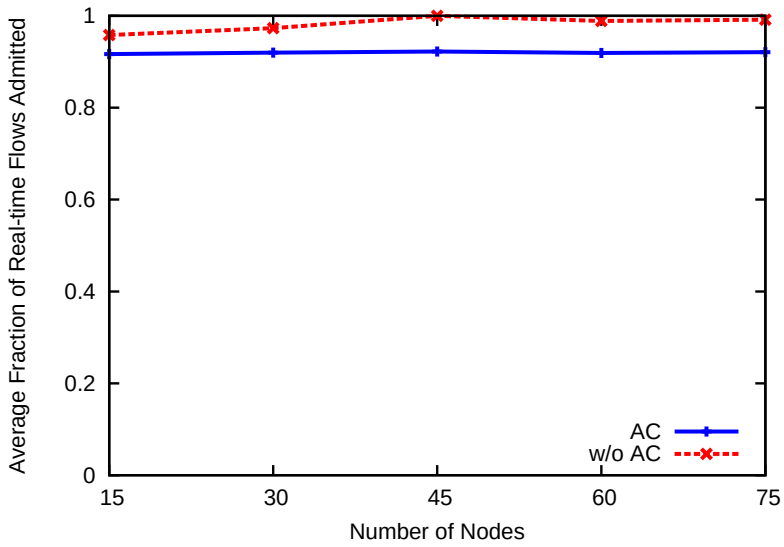

Figure 5. Comparison of fraction of the real-time flows admitted by the admission controller and the fraction of the flows performed on time in the absence of the controller for different number of nodes in the DTN.

It is worth mentioning that our simulation study of the resource utilization in DTNs reports significantly small values (less than 5%) for the average utilizations of resources (i.e. communication links) in the network. The reason lies in the spacial unavailability of resources which is the main characteristic of DTNs and results in network utilization being degraded dramatically compared to regular distributed systems.

Figure 6 presents the average ratio of end-to-end delay to delay bound computed based on the theorem. The figure quantifies pessimism of our approach more precisely and ensures that our approach does not suffer from a significant degree of pessimism. It also confirms that the technique enjoys considerable scalability properties and overestimation of the worst-case end-to-end delay does not increase with the network size.

The fraction of the real-time flows admitted or performed on time versus deadline resolution is plotted in figure 7. As the value of deadline resolution increases, end-to-end deadline of flows become more relaxed and a larger fraction of real-time flows is admitted. As the figure shows, the gap between AC and w/o AC is less than 8% which presents an appreciable performance compared to that of the w/o AC approach that can be deemed as an optimal admission controller.

The comparison of the average ratio of end-to-end delay to computed delay bound versus deadline resolution is depicted in figure 8. The figure reports that the pessimism of our approach does not grow with increase/decrease in the value of the deadline resolution and its effectiveness remains nearly constant.

6. Conclusions

In this paper, we developed a framework to capture timing properties of recurrent disruption-tolerant networks, and provided a delay composability rule for DTNs. The work considers a generalized workflow which involves interactions with physical entities both in time and in space. The distributed systems we are concerned with follow the communication paradigm of DTN-like systems and violate the fundamental assumption of preceding works in literature that system resources are always available in space (although not available in time). Specifically, in this kind of distributed systems there exists two types of resource unavailability that are addressed: the resource may not be available in time (i.e. higher priority packets are being transferred through the encounter), and the resource may not be available in space (i.e. the next custodian is not in range yet).

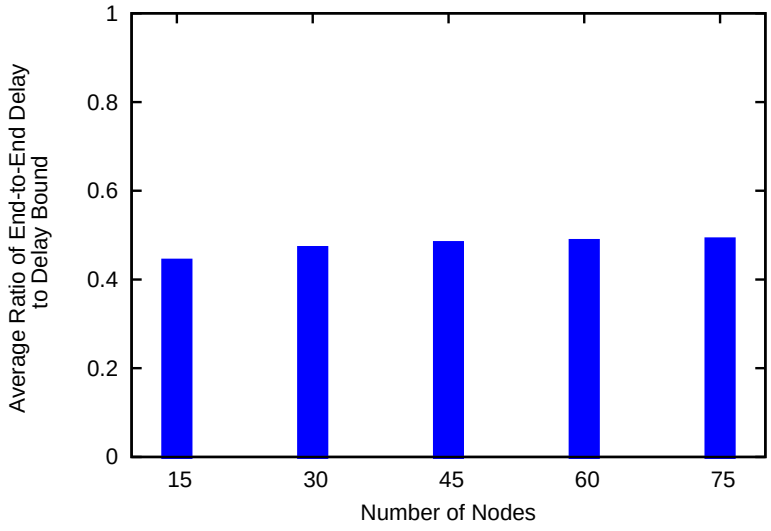

Figure 6. Comparison of average ratio of end-to-end delay to estimated delay bound using the delay theorem for different number of nodes in the DTN.

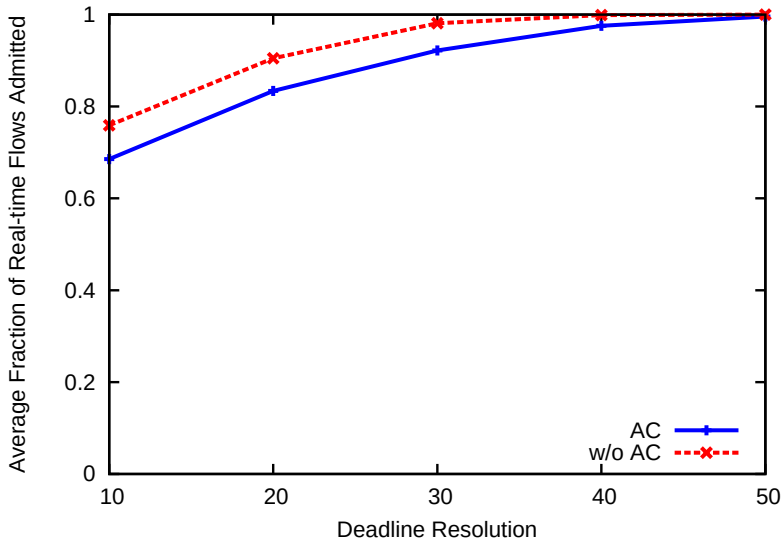

Figure 7. Comparison of the fraction of the real-time flows admitted by admission controller and the fraction of the flows performed on time in the absence of the controller for different values of deadline resolution.

We provided an estimation of the worst-case end-to-end delay of data flows in DTNs and analyzed schedulability of the distributed workload. Our approach reduces the entire disruption-tolerant network into a single virtual node, the workload of which imposes end-to-end delays no less than those of the original DTN. Thus a uniprocessor scheduling analysis can be employed to determine whether or not the original multi-criticality spatially-distributed workload is feasible. We evaluated our approach and showed that the approach has moderate pessimism and can be employed as an effective admission controller filtering out excess of the workload which may not be feasible prior to workload setup in DTNs.

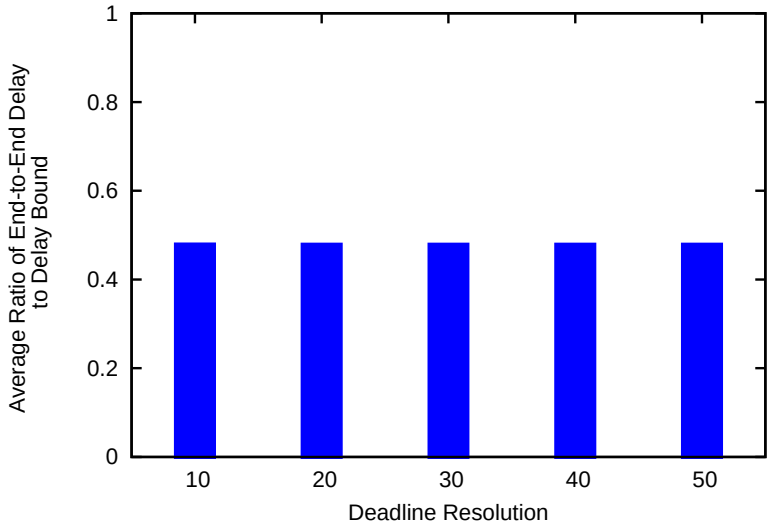

Figure 8. Comparison of average ratio of end-to-end delay to estimated delay bound using the DTN delay theorem for different values of deadline resolution.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *ACM SIGCOMM*, 2003.

[2] http://www.dtnrg.org.

[3] P. Jayachandran and T. Abdelzaher, "Delay composition algebra: A reduction-based schedulability algebra for distributed real-time systems," in *RTSS*, December 2008, pp. 259–269.

[4] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *ACM SIGCOMM*, 2004.

[5] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, Tech. Rep., 2004.

[6] A. D. Nicolo and P. Giaccone, "Performance limits of real delay tolerant networks," in *IEEE WONS*, 2008.

[7] C. Liu and J. Wu, "Routing in a cyclic mobispace," in *ACM MobiHoc*, 2008.

[8] C. Liu, J. Wu, and I. Cardei, "Message forwarding in cyclic mobispace: the multi-copy case," in *IEEE MASS*, 2009.

[9] D. Hay and P. Giaccone, "Optimal routing and scheduling for deterministic delay tolerant networks," in *IEEE WONS*, 2009.

[10] M. Y. S. Uddin, H. Ahmadi, T. Abdelzaher, and R. Kravets, "A low-energy, multi-copy inter-contact routing protocol for disaster response networks," in *IEEE SECON*, 2009.

[11] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: the single-copy case," in *IEEE/ACM Transactions on Networking*, 2008.

[12] ——, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," in *IEEE/ACM Transactions on Networking*, 2008.

[13] J. Xu and D. Parnas, "On satisfying timing constraints in hard real-time systems," *IEEE Transactions on Software Engineering*, vol. 19, no. 1, pp. 70–84, January 1993.

[14] G. Fohler and K. Ramamritham, "Static scheduling of pipelined periodic tasks in distributed real-time systems," in *Euromicro Workshop on Real-Time Systems*, June 1997, pp. 128–135.

[15] B. Kao and H. Garcia-Molina, "Deadline assignment in a distributed soft real-time system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 12, pp. 1268–1274, 1997.

[16] Y. Zhang, C. Lu, C. Gill, P. Lardieri, and G. Thaker, "End-to-end scheduling strategies for aperiodic tasks in middleware," University of Washington at St. Louis, Tech. Rep. WUCSE-2005-57, December 2005.

[17] M. D. Natale and J. A. Stankovic, "Dynamic end-to-end guarantees in distributed real time systems," in *IEEE Real-Time Systems Symposium*, 1994, pp. 215–227.

[18] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Elsevier Microprocessing and Microprogramming*, vol. 40, no. 2-3, pp. 117–134, 1994.

[19] J. Palencia and M. Harbour, "Offset-based response time analysis of distributed systems scheduled under edf," in *Euromicro Conference on Real-Time Systems*, July 2003, pp. 3–12.

[20] R. Pellizzoni and G. Lipari, "Improved schedulability analysis of real-time transactions with earliest deadline scheduling," in *RTAS*, March 2005, pp. 66–75.

[21] R. Cruz, "A calculus for network delay, part i: Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, January 1991.

[22] ——, "A calculus for network delay, part ii: Network analysis," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, January 1991.

[23] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *IEEE International Symposium on Circuits and Systems*, vol. 4, May 2000, pp. 101–104.

[24] B. Jonsson, S. Perathoner, L. Thiele, and W. Yi, "Cyclic dependencies in modular performance analysis," in *ACM EMSOFT*, Oct. 2008, pp. 179–188.

[25] E. Wandeler, A. Maxiaguine, and L. Thiele, "Quantitative characterization of event streams in analysis of hard real-time applications," in *IEEE RTAS*, May 2004, pp. 450–459.

[26] P. Jayachandran and T. Abdelzaher, "A delay composition theorem for real-time pipelines," in *ECRTS*, July 2007, pp. 29–38.

[27] P. Jayachandran and T. Abdelzaher, "Delay composition in preemptive and non-preemptive real-time pipelines," *Invited to Real-Time Systems Journal: Special Issue on ECRTS'07*, vol. 40, no. 3, pp. 290–320, December 2008.

[28] P. Jayachandran and T. Abdelzaher, "Transforming acyclic distributed systems into equivalent uniprocessors under preemptive and non-preemptive scheduling," in *ECRTS*, July 2008, pp. 233–242.

[29] P. Jayachandran and T. Abdelzaher, "End-to-end delay analysis of distributed systems with cycles in the task graph," in *ECRTS*, July 2009.

[30] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[31] E. Bini, G. Buttazzo, and G. Buttazzo, "A hyperbolic bound for the rate monotonic algorithm," in *13th ECRTS*, June 2001, pp. 59–66.

[32] A. N. Audsley, A. Burns, M. Richardson, and K. Tindell, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering*, vol. 8, no. 5, pp. 284–292, 1993.

[33] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *RTSS*, December 1989, pp. 166–171.