Prioritized Data Synchronization for Disruption Tolerant Networks

Jiaxi Jin, Wei Si, David Starobinski, Ari Trachtenberg Dept. of Electrical and Computer Engineering
Boston University
jin@bu.edu, weisi@bu.edu, staro@bu.edu, trachten@bu.edu

Abstract

We consider the problem of synchronizing prioritized data on two distinct hosts in disruption-tolerant networks (DTNs). To this effect, we propose and analyze a new interactive protocol for priority-oriented synchronization, called P-CPI, that is especially efficient in terms of bandwidth usage. This middleware protocol has features that are particularly useful for DTN routing in constrained or tactical environments, including (i) communication and computational complexity primarily tied to the number of differences between the hosts rather than the amount of the data overall and (ii) a memoryless fast restart after interruption. We provide a novel analysis of this protocol, substantiating a high-probability performance bound and memoryless fast-restart in logarithmic time. As a proof of concept, we demonstrate improved delivery rate and reduced metadata and average delay in a DARPA-supported DTN routing application called RAPID.

A version of this paper appeared as:

- J Jin, W Si, D Starobinski, A Trachtenberg, *Prioritized data synchronization for disruption tolerant networks* IEEE MILITARY COMMUNICATIONS CONFERENCE (MILCOM) 2012, October 2012.

I. Introduction

A. Background

In disruption-tolerant networks (DTNs), also known as delay-tolerant networks, mobile nodes (not only end-systems) experience sporadic connectivity access to the rest of the network [1]–[3]. Examples of such networks include those operating in tactical environments, where disruptions occur due to the lack of infrastructure.

One of the key issues in DTNs is how to route a packet towards its destination when, at any given point of time, the network is not fully connected. In order to speed up delivery and ensure reliability, it has been suggested that the same packet should be transferred to several different nodes, thus creating several replicas (see, e.g., [3], [4]).

The problem with the replication approach is that the number of replicas can potentially become very large, thus wasting precious communication and storage resources and severely degrading performance [4]. Within such a context, a communication-efficient data synchronization protocol is essential to ensure that nodes exchange only the replicas that they do not already possess.

Furthermore, due to the limited bandwidth (and time) available at each meeting, two nodes may only be able to exchange a subset of their differing packets. In such cases, it is critical that highest priority data be forwarded first. Thus, several DTN protocols [4], [5] assign priorities to packets based on metadata information, such as packet importance, delivery deadlines, and statistics of inter-meeting times between different pair of nodes. In essence, packets with higher priority are transmitted first. However, these works do not explicitly address the problem of how to efficiently reconcile the data sets of the two hosts.

B. Contributions

In this paper, we propose a new synchronization middleware for reconciling two remote prioritized sets of data in DTNs, based on a protocol called Priority-based Characteristic Polynomial Interpolation

(P-CPI). We conduct a worst-case and high-probability analysis of P-CPI and prove that if the number of differences between the data sets of two hosts is m, then both the communication complexity and the computational complexity are $O(m \log m)$ with *high-probability* (i.e., with probability approaching one as m gets large). Further, we prove that P-CPI can be stopped and restarted with minimal overhead, meaning there is no need to keep state information between any two meetings, and we derive bounds on the computational overhead of such restarts. Finally, as a proof of concept, we demonstrate P-CPI as a synchronization middleware, for the RAPID DTN routing protocol [4]. Simulations for typical DTN settings demonstrate the potential of sizable improvement for various performance metrics, including metadata overhead, delay and delivery rate.

C. Related Work

The literature offers several approaches for data synchronization, although we wish to stress that none of them appear to consider prioritization for their data. The most related work to our synchronization algorithm are based on mathematical synchronization of data [6]–[9], which are described in detail in Section II. The work in [7] describes the approach of characteristic polynomial interpolation, which has nearly-optimal communication complexity, and [8] outlines an interactive scheme for synchronization together with a *worst-case* and *average-case* analysis. Our work contributes a *high-probability* analysis of the communication complexity, which is shown to coincide with the average-case performance.

Next, we briefly discuss related work in DTN routing. The RAPID [4] DTN routing protocol, which serves as our proof of concept example, can be configured to minimize average packet delay, maximize average delivery rate and minimize the maximum delay of all packets. Nodes prioritize packets by their utility functions, which are calculated in terms of these metrics. Experiments in [4] show that under conditions such as high packet generation rates and small packet sizes, the fraction of metadata sent may be quite significant. This fact motivates us to propose schemes to efficiently manage the exchange of such metadata. Based on the same testbed, a prioritized DTN routing protocol MAXPROP [10] is established and tested; in this protocol, prioritized packet transmission is based on history and saved intermediate results, but this requires additional memory and therefore only works well for relatively static network.

PREP [5] is another routing protocol that prioritizes packets, according to the estimated cost from the current node to the destination. Transmission cost is estimated between each pair of nodes. In this manner, the network is seen as a graph. The utility function of a packet is equivalent to the cost of the shortest path in the graph (based on the information available at the node). In PROPHET [2], each node keeps a vector of *delivery predictability*, one entry for each destination. A node decides to transfer a packet to the other node if the delivery predictability of the destination of the packet at the other node is higher than its own. The metadata exchanged include the vectors of delivery predictability.

D. Outline

The rest of this paper is organized as follows. Section II provides abstract descriptions of Characteristic Polynomial Interpolation (CPI)-based approaches [7]–[9] to data synchronization, which are the building blocks of our protocol, and then presents our new protocol. We describe the Priority-based CPI (P-CPI) algorithm to handle partial and priority-based synchronization and conduct a detailed analysis of its performance. Section III discusses the implementation of P-CPI into RAPID and shows simulation results. Section IV concludes the paper.

II. Prioritized Set Reconciliation

A. The set reconciliation problem

The basic model of the reconciliation problem is as follows. A local host A and another remote host B possess sets S_A and S_B respectively. While neither of hosts is assumed to know the contents of the other host's set in advance, their goal is to compute the *symmetric difference* between two sets using minimum

amount of communication. The *symmetric difference* of sets S_A and S_B is defined as the set of elements which are in either of the sets but not in their intersection, i.e., $S_A \oplus S_B = (S_A - S_B) \cup (S_B - S_A)$. For the purpose of analysis, we assume that the element of the sets are all b-bit numbers (in practice this can be done by hashing), which bounds the size of the symmetric difference between two sets to 2^b elements.

B. CPIsync

The work in [7] presents a *Characteristic Polynomial Interpolation*-based synchronization algorithm (dubbed CPIsync [11]) that translates the set reconciliation problem into the problem of rational function interpolation. More precisely, given two sets of b-bit numbers S_A and S_B respectively, this algorithm can synchronize the two sets using one message of $S_A \oplus S_B$ samples. The algorithm is only logarithmically dependent on the sizes of the sets (i.e., its complexity is proportional to b). Thus, two data sets could each have millions of entries, but if they differ in only m of them, then each set can be synchronized with the other by a single round communication using one message whose size is about that of mb bits (i.e. the number of differences multiplied by the number of bits per set element).

The *characteristic polynomial* of set $S = \{x_1, x_2, ..., x_n\}$ is defined as

$$P_S(Z) = \prod_{i=1}^{n}(Z - x_i). \tag{1}$$

The translation of the data into characteristic polynomials is the key to CPIsync algorithm. During the synchronization, one host sends sampled values of its characteristic polynomial to the other host; the number of samples must be at least as many as the number of symmetric differences (i.e. m) between the two hosts. The second host then computes the differing entries by interpolating the rational function corresponding to the ratio of the two characteristic polynomials from the received samples.

CPIsync requires hosts to have a bound $\bar{m}$ on the number of symmetric differences m between sets S_A and S_B to know how many samples must be communicated between them. In other words, one is assured to recover the symmetric difference of size $|S_A \oplus S_B|$ if it is no larger than $\bar{m}$.

Protocol CPIsync$(S_A, S_B, \bar{m})$: Set Reconciliation for S_A and S_B with at most $\bar{m}$ differences.
1. Each host evaluates its *characteristic polynomial* at $\bar{m}$ sample points.
2. Host A sends the $\bar{m}$ evaluations of S_A to Host B.
3. By characteristic polynomial interpolation, host B computes the set differences, $\Delta A = S_A - S_B$ and $\Delta B = S_B - S_A$.
4. Host B sends the result back to Host A.

As given in [7], CPIsync has a communication complexity of $\Theta(\bar{m}b)$. The main bottleneck of CPIsync is its computation complexity, which is $\Theta(b\bar{m}^3 + bmk)$, cubic in the upper bound $\bar{m}$. This computation cost would be unnecessarily expensive when the upper bound guess $\bar{m}$ on the symmetric difference m is conservative (i.e. $\bar{m} \gg m$).

C. Protocol: Priority CPI (P-CPI)

In this section, we propose a new protocol, called Priority-based CPI (P-CPI), to support efficient prioritized data synchronization in DTNs. P-CPI uses a "divide-and-conquer" approach to handle prioritization (a similar approach named I-CPI appeared in our previous work [8]). More precisely, in P-CPI, set elements are first split by priority, then the synchronization runs on each pair of subsets in decreasing order of priority, which guarantees that the limited network bandwidth is first used for data entries with high priority. If the number of elements of same priority is too large for a single CPIsync to solve,

recurring partition based on number field are performed until CPIsync succeeds on every paired subsets. The partitioning process of the original set can be symbolized by a data structure so-called *partition tree*, in which each node represents a (sub)set of elements, and the process of synchronization is essentially a depth-first traversal on a partition tree .

D. Worst-case analysis

Our analysis on P-CPI makes use of some common notation. First of all, our set elements are represented by b-bit vectors. The *difference bound* $\bar{m}$, is the designed upper bound of the size of the symmetric difference that can be determined by one call of CPIsync. Similarly, the *partition factor* p represents the number of children that an internal node of our partition tree can have. The *priority ratio* η is the ratio of the number of differences (between the two sets being synchronized) that are at high priority to the total number of differences. If there are more than two priorities in the system, η represents the ratio of number of differences with priority above a given threshold to the total number of differences. Note that $\eta = 1$ for the worst case when all the differences are of high priority and need to be synchronized. Finally, $I(\eta m)$ is the overall number of invocations to CPIsync during the execution of P-CPI on two sets with ηm symmetric differences.

The following Lemma provides a worst-case upper bound on the number of CPIsync calls by P-CPI.

Lemma 1: For P-CPI to synchronize two sets with m symmetric differences, the number of invocations of CPIsync is bounded by

$$I(m) \leq 1 + \frac{m}{\bar{m}} p \lceil \log_p(2^b) \rceil \tag{2}$$

Proof: The work in [8] bounds the number of invocations of CPIsync by

$$I(m) \leq 1 + \frac{m}{\bar{m}} p \lceil \log_p s \rceil \tag{3}$$

as the worst-case condition for I-CPI, where s stands for set size. Substituting $s = 2^b$ gives the desired bound. ■

Given that $\Theta(b\bar{m}^3 + bmk)$ is the computation complexity of CPIsync [7], multiplied by (2) we attain the *worst-case computation complexity* of P-CPI as $\Theta(m\bar{m}^2 b^2 \frac{p}{logp})$.

Similarly, the *worst-case communication complexity* of P-CPI is $\Theta(m\frac{p}{\log p}b^2)$ given that the communication complexity of CPIsync is $\Theta(mb)$.

E. High-probability analysis

In this subsection, a new probabilistic analysis shows that the number of CPIsync invocations is $O(\eta m \log(\eta m))$ with high-probability, which is particularly needed when there is a large gap between the expected-case (mb) and worst-case (mb^2) [8]. Since a set element can be represented by any b-bit string, we assume a uniform-random distribution of the symmetric differences between sets, which is so implemented by a pseudo-random hashing of the data before the synchronization. Based on this assumption, we derive a high-probability bound on the number of CPIsync calls by P-CPI. The analysis resembles that of quicksort [12], but the partitioning process of the tree is different, thus requiring a different analysis. The proof of the following theorem is abridged due to space limitation.

Theorem 1: For P-CPI to synchronize two sets with ηm uniform-randomly distributed differences in total, the number of calls to CPIsync is $O(\eta m \log(\eta m))$, with probability at least $1 - \frac{1}{\eta m}$.

Proof: Let $m' = \eta m$ and, for the sake of exposition, let the partition factor $p = 2$. Other partition factors would simply change the base of our logarithms in the proof, and would not affect the conclusion.

In a binary partition tree, choose any root-to-leaf path P. The Chernoff-Hoeffding bounds [12] helps to validate that

$$Pr\Big[|P| > 4\log_2 m'\Big] < \frac{1}{m'^2}, \tag{4}$$

where $|P|$ is the length of the chosen root-to-leaf path P, i.e. the total number of nodes along it.

Then a simple application of union bound shows:

$$Pr\left[\exists P, |P| > 4\log_2 m'\right] \leq \frac{1}{m'}, \tag{5}$$

provided that the number of leaf nodes in a partition tree is $O(m')$. So the overall number of nodes in the partition tree, which also represents number of calls of CPIsync, $I(m')$, is:

$$I(m') \in O(m' \log m') \tag{6}$$

with probability at least $1 - \frac{1}{m'}$ ■

Once again, by multiplying the number of CPIsync invocations in the high-probability case with CPIsync's basic communication and computation complexity, we have that the *high-probability computation complexity* of P-CPI is $I(m')\Theta(b\bar{m}^3 + bmk) \in O(\eta mb(\bar{m}^2 + k)\log(\eta m))$ and the *high-probability communication complexity* of P-CPI which is $I(m')(\bar{m}b + 2) \in O(\eta mb \log \eta m)$ both with probability at least $1 - \frac{1}{\eta m}$.

For comparison purpose, we assume that $\bar{m}$ and p are constants given a-priori. Thus the number of CPIsync invocations is $O(mb)$ in the worst-case and $O(m \log m)$ in the high-probability case. Since m cannot exceed 2^b by definition, and in practice m is often much smaller than 2^b, we then conclude that, in many applications, the high probability performance of P-CPI is markedly better than its worst-case performance.

F. Restarting interrupted P-CPI

A commonly-used technique for restarting a synchronization is to save the intermediate results at the interruption, and then reload them when the synchronization process resumes. However, in distributed systems such as DTNs, the memory capacity at each node is usually limited and therefore not capable of saving too many intermediate results.

P-CPI enables a *memoryless* fast restart of previously interrupted synchronization, and no modification to the original P-CPI is needed for this feature. By using P-CPI, a node in a mobile network may use exactly the same protocol to synchronize with any node it meets, and has a fast restart if i) the last synchronization between the same hosts was interrupted and ii) no new difference was added since then.

The execution of P-CPI is essentially a depth-first traversal of one or more pairs of partition trees. If the execution is interrupted and the synchronization breaks at a certain pair of nodes (i.e. a *break pair*), we can conclude that there must be unsynchronized pair(s) at positions not earlier than the break pair in the depth-first traversal of the same pair of partition trees. We call a pair of nodes *synchronizable* if they are unsynchronized and contains no more than $\bar{m}$ differences. In the case of an interruption, hosts update their databases with symmetric differences determined before the interruption and wait for a restart. By restarting P-CPI between the same hosts later (assuming no new differences are added), the synchronization will resume once it finds the first synchronizable pair in their partition trees.

Theorem 2: The number of CPIsync calls before a synchronizable pair is found in a pair of p-ary partition trees with bit-string length b is upper-bounded by bp.

Proof: The algorithm proceeds according to the following flow until it finds a synchronizable pair:

- if the CPIsync call on a pair of nodes fails to find differences, which means at least one synchronizable pair are their descendants, then the execution proceed to the first left child nodes of the pair;
- else if the CPIsync call returns no differences (which means no synchronizable pair exists in their descendants) and at least one node in the pair has a non-empty right sibling, then the execution proceed to their (first) right siblings,
- otherwise (when no differences are found and no right sibling exists), the algorithm returns and reports that the current pair of partition trees are fully synchronized.

Since a p-ary partition tree recursively partitions the field of range 2^b equally into p partitions and each leaf nodes is on a (sub)field of range at least (some constant) $\bar{m}$ differences, the height of the partition tree is at most b. The proceeding flow described above attains at most p CPIsync calls per level (as it proceeds along p siblings). Thus the number of CPIsync calls is at most bp. ■

As an example, consider two sets containing one million elements and using P-CPI with a partition $p = 2$. In that case, the number of CPIsync calls before a synchronizable pair is found is at most 40, no matter what is the number of differences between the two sets, how many of these differences were previously reconciled, and the value of the parameter $\bar{m}$ used in each CPIsync call. Further, no special information needs to be maintained by the nodes for P-CPI to achieve this performance.

III. Numerical Results

In this section, we present numerical results illustrating the performance of P-CPI. We also describe our implementation of P-CPI into RAPID and report the improvements observed from simulations. We use RAPID as a proof of concept because it is DARPA-supported and has been shown to compare favorably to several other DTN routing protocols.

A. Modified RAPID with P-CPI Deployed

As pointed in [4], RAPID requires a full synchronization of metadata ahead of any packet transmission. When the size of the metadata is large, this approach bears the risk of limiting the amount of useful information (i.e., packets) that can be exchanged at a meeting between two nodes. The authors of [4] left this problem open for future work.

P-CPI provides a good strategy to address this problem. Using P-CPI, metadata and packets with high priority are sent first. Then, if the link is still available, lower priority information is carried over the link. Therefore, we propose a modified version of RAPID with P-CPI deployed. In the modified protocol RAPID-PCPI, the estimated benefit of replicating a certain packet is calculated by the same metadata-based utility function as in RAPID and the high or low priority of a packet is then determined by a threshold of expected benefit. Packets with higher expected benefit evaluations are assigned high priority while the rest are arranged to go with low priority. In our implementation, the numbers of low and high priority packets are set equal. However, the fraction of high priority packets can also be made application dependent and dynamic, if desired. We also note that a packet and its replicas could be assigned different priorities at different nodes. However, a higher level protocol can detect this when metadata for the packet are reconciled and avoid the need of transmitting the packet itself.

Protocol RAPID-PCPI(X, Y):
0. *Initialization:* Establish the partition tree of packets with binary priorities at node X and Y.
1. *Synchronization (high priority):* Synchronize metadata of packets of high priority with Y.
2. *Packet Delivery (high priority):* Deliver/Replicate packets of high priority.
3. *Synchronization (low priority):* Synchronize metadata of packets of low priority with Y.
4. *Packet Delivery (low priority):* Deliver/Replicate packets of low priority.
5. *Termination*: End transfer when out of radio range or all packets replicated.

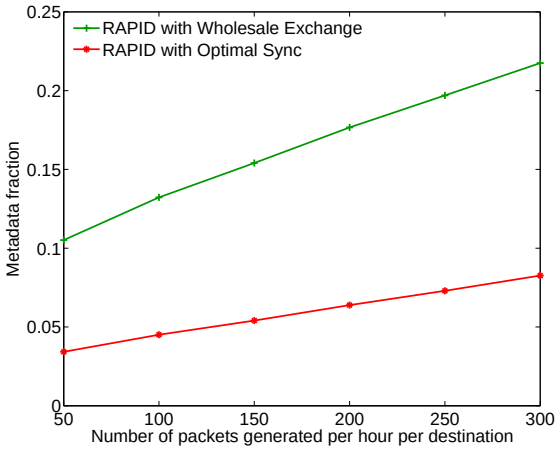


Fig. 1. Fraction of metadata versus packet generation rate

B. Performance Evaluation

We next illustrate through simulation the performance of RAPID, with and without P-CPI being deployed. As a baseline, we assume that the original RAPID protocol uses an "oracle" (which is unfeasible in practice) to determine in advance the list of differing metadata entries. We refer to this approach as *Optimal Sync*. As a baseline, we also consider the case where RAPID uses *Wholesale Exchange*, i.e., exchanges metadata in wholesale, during each synchronization. The simulations are conducted with the RAPID simulator developed by the authors of [4]. The mobility model in the simulation is obtained from a vehicular DTN testbed, DieselNet.

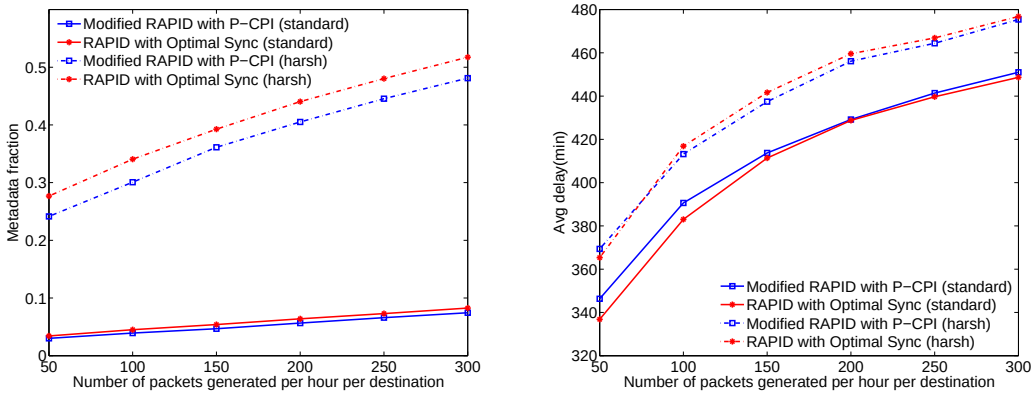
The default parameters for the simulation are listed in Table I, most of which are the same with simulation setup in [4].

Number of nodes	max of 40
Buffer size	400MB
Average transfer opp. size	given by real transfers among buses
Duration	19 hours each trace
Size of a packet	1 KB
Average bandwidth	400Kb/s
Packet generation interval	1 hour
Optimization metric	average delay
Number of priority levels	2

TABLE I
EXPERIMENT PARAMETERS

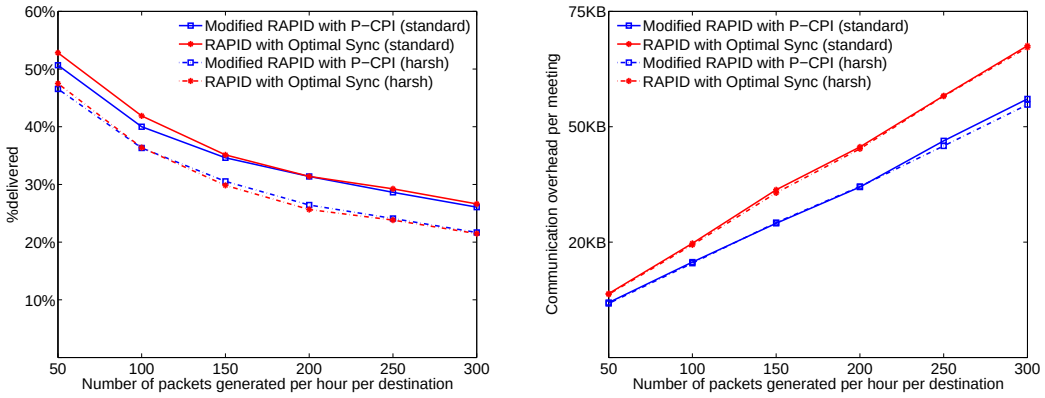
The metrics we use to evaluate the system performance include average delay, delivery rate, metadata fraction and communication overhead. More specifically, the delay of a delivered packet is defined to be the duration from its generation to delivery and the delay of undelivered packet is the duration from its generation to the end of simulation.

We first run simulations using the default settings shown in Table I. Figure 1 shows the metadata fraction, i.e., the ratio of exchanged metadata to all exchanged data in the network, as a function of the packet generation rate. The figure illustrates the problem left open by the authors of [4], exchange of metadata could potentially jam the network traffic (and an optimal synchronization protocol can effectively reduce this effect). In fact, at high packet generation rate, RAPID with P-CPI (which is proved to be nearly-optimal in communication) yields about a three-fold reduction in the metadata fraction.



(a) Fraction of exchanged metadata to all data sent in the network

(b) Average packet delivery time (delay from generated to delivered/expired)



(c) Percentage of packets delivered before expired to all packets generated

(d) Average communication overhead per meeting

Fig. 2. Performance contrast of two protocols: different features (specified in subfigures) versus packet generation rate, using two different settings

We present P-CPI as a practical synchronization middleware in DTN routing protocols such like RAPID. We next show the detailed comparison between RAPID using Optimal Sync and P-CPI and the additional benefits brought by prioritization. We run simulations in two different scenarios, a *standard* scenario which inherits the default settings in Table I, and another *harsh* scenario with the average packet size set to 100 bytes and the average bandwidth set to 40 Kb/s, the result is shown in Figure 2.

Figure 2(a) shows that the metadata overhead becomes much more significant in such situations. As a result, the metadata reduction achieved with P-CPI results in major performance gains. The reason of P-CPI performs even better than Optimal Sync is that Optimal Sync initially performs an optimal synchronization of the entire metadata, while P-CPI first synchronizes high priority metadata and packets and then, if time is still available, synchronizes low priority metadata and packets. This reduction in the metadata fraction can lead to an improvement of about 5% in the percentage of packets delivered. Thus, Figure 2(b) shows that P-CPI stays closed to optimal with standard settings in the average delivery time (delay), and further reduces it and outperforms the Optimal Sync in harsh scenario. Similarly, Figure 2(c) indicates corresponding improvement for the percentage of packets delivered within the given deadline. Figure 2(d) demonstrates the effectiveness of P-CPI in sense of communication overhead. Again the reason why the communication overhead of P-CPI is even better than RAPID of Optimal Sync oracle is P-CPI sends not all but only part of the metadata at first. The simulation results in Figure 2 demonstrate that the newly proposed modified RAPID with P-CPI performs nearly the same as the original RAPID with

Optimal Sync with standard settings and even better in harsh scenario.

IV. Conclusion

In this paper, we have introduced, analyzed, and simulated a new synchronization middleware for DTNs, called Priority-based Characteristic Polynomial Interpolation (P-CPI). Specifically, we have provided novel worst-case and high-probability performance analysis of the computation and communication complexity of P-CPI. Our simulations demonstrated that the computation and communication complexity of P-CPI grows close to linearly with the number of differences, depending only weakly (i.e., logarithmically) on the number of elements in the sets. These complexities also scale proportionally to the desired priority ratio. We have also proven that P-CPI can be stopped and quickly restarted at any time without incurring any extra memory overhead, a feature that is particularly useful in networks with a large number of nodes.

In addition to the analysis, we have demonstrated the practical benefit of using P-CPI in a DTN setting by implementing it as a synchronization conduit for the well-established *RAPID* DTN routing protocol. Simulations obtained using the original RAPID simulator show that P-CPI leads to significant reduction in the communication overhead of metadata compared to a simple wholesale transfer approach, hence solving a problem left open by the authors of RAPID. As a practical approach to synchronization, P-CPI yields near optimal performance in the average delivery time of packets and other related metrics. We expect that P-CPI could serve as an effective synchronization middleware for other DTN routing protocols beside RAPID, an interesting area left for future work.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 27–34.

[2] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, pp. 19–20, July 2003.

[3] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, 2005, pp. 109–120.

[4] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *IN PROC. ACM SIGCOMM*. ACM, 2007, pp. 373–384.

[5] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan, "Prioritized epidemic routing for opportunistic networks," in *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, ser. MobiOpp '07. New York, NY, USA: ACM, 2007, pp. 62–66.

[6] K. A. S. Abdel-Ghaffar and A. El Abbadi, "An optimal strategy for comparing file copies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp. 87–93, January 1994.

[7] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Trans. on Information Theory*, vol. 49, 2003.

[8] Y. Minsky and A. Trachtenberg, "Scalable set reconciliation," in *40th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2002.

[9] D. Starobinski, A. Trachtenberg, and S. Agarwal, "Efficient pda synchronization," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 41–50, 2003.

[10] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –11.

[11] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for pdas and mobile devices," *IEEE Network*, vol. 16, pp. 22–28, 2002.

[12] D. P. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomised Algorithms*. Cambridge University Press, 2009, pp. 46-47.