# Data to the People

Arseny Kurnikov
Aalto University, Comnet
Espoo, Finland
arseny.kurnikov@aalto.fi

Teemu Kärkkäinen
Aalto University, Comnet
Espoo, Finland
teemuk@netlab.tkk.fi

Jörg Ott
Aalto University, Comnet
Espoo, Finland
jo@netlab.tkk.fi

## ABSTRACT

Delay Tolerant Networking (DTN) and Information Centric Networking (ICN) bear numerous similarities. In particular, both may operate on larger information units than packets that can be identified by means of metadata. Caching, searching, and content-based routing have been explored individually in both areas. In this paper, we combine the two techniques and extend the traditional caching and data retrieval approaches ("read") to support changing data in caches ("write") in order to allow local content copies to evolve through modifications during network partitions.

## Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design; Network Protocols

## Keywords

DTN; Content-Based Processing; Data Sharing Application

## 1. INTRODUCTION

Delay-tolerant Networking (DTN) [5] relaxes several of the assumptions on the underlying connectivity inherent to the Internet. In particular, DTN does not assume low RTTs between communicating nodes nor an end-to-end path at any single instant in time. Routing decisions may be deferred if no suitable next hop is found when a message is received, and the message may be stored for an extended period of time until a path is found. DTN messages may be of arbitrary size [2, 12] which allows conveying self-contained application data units that include, e.g., context, metadata, security credentials, etc. along with the data. Thereby, the number of round trips to carry out protocol operations can be minimized [7].

An important feature of self-contained messages is that nodes along the path have all necessary information at hand to provide supportive functions without having to manage flow states or gather content as in the Internet. Suggestions for such support are manifold but, in particular, *information-centric networking (ICN)* concepts were devised early on for DTNs, including *intentional naming* [4] or *content-based routing* (e.g., [4, 3], opportunistic caching [9], and searching for content [8, 6]. Recent research has begun merging the highly related ideas of DTN and ICN concepts: e.g., running DTN protocols as a robust network layer services in NetInf or building inclusive *lowest common denominator networks* [10]. ICN/DTN functions retrieving cached content copies may alter a request message or metadata en route, but they are mostly *read* operations concerning the actual message content, with the exception of creating transcoded message *copies* to serve specific devices.

In this paper, we seek broadening the ICN/DTN concepts and explore *write* operations on content inside the network. We consider a self-contained message carrying a content item that represents an instance of an application object in flight. With basic caching, an exact copy is created to serve further requests, which keeps content available even when the origin server is unreachable. Going beyond passive retrieval allows for more sophisticated operations in the absence of connectivity.

We start by looking at the specific case of **Network-based distributed applications**: A cached copy of a message becomes mutable. Instead of retrieving an object and then uploading a modified copy, we allow its modification. The cached object is updated according to the modification information carried in the update message and the message is forwarded towards the origin server and possibly to other nodes holding copies of the object.

## 2. SYSTEM DESCRIPTION

We design our bundle processing framework based upon the bundle protocol query extension and provide an initial implementation for the IBR-DTN router. Its modular structure supports our two extensions well: one routing extension to support Bundle Protocol Queries and one API extension to provide an application access to the added functionality.

### 2.1 Bundle Protocol Queries

The Bundle Protocol Query (BPQ) extension block allows explicitly identifying the requested resource name in a header when a node requests the resource from the destination. The extension block comprises three fields: 1) a BPQ kind describing if the bundle is a request or a response; 2) a BPQ value containing the name of the requested resource in case of a querying bundle and the name for the resource in the payload for responses; and 3) an original timestamp of the bundle creation.

When a bundle with a request BPQ extension arrives at a BPQ-aware node, the node checks its storage for BPQ responses with the matching BPQ values. If a response is found a copy of the response bundle is sent back to the requesting node and the request bundle can be dropped. Otherwise, the request is queued for forwarding. Incoming responses are checked against queued requests, enqueued for forwarding and possibly inserted into a separate cache.


*CHANTS'13,* September 30, 2013, Miami, Florida, USA.
ACM 978-1-4503-2363-5/13/09.
http://dx.doi.org/10.1145/2505494.2505505.

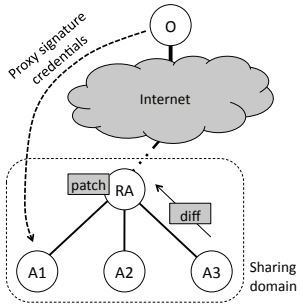


**Figure 1: Basic operation of router-based object modifications**

Figure 1 depicts a scenario for enabling document modification in the router cache, where `O` is the origin server, `RA` is the caching router and `A`'s are clients. To support "write" operations, we define a new BPQ kind: update. When a node makes modifications to the document it generates a bundle containing the changes to the document, sets the BPQ-kind to "update" and modifies the BPQ-value to include the new revision number of the original document and a URI parameter indicating the patch operation type. When receiving such an update, a BPQ-aware node searches for an older response with the corresponding BPQ value and applies the patch to keep it up-to-date. The update messages are forwarded towards the content origin. Incoming updates are processed in an FCFS order and identifiable conflicts lead to a BPQ "error" message returned to the initiator of the conflicting operation.

## 2.2 Document sharing

This framework offers the mechanisms for building document sharing applications. We are moving the user documents managed in the cloud topologically closer to those parts of the network with nodes that want to access and modify the documents, taking the data *closer to the people*, which is especially useful when the cloud access is intermittent and subject to delays.

The origin keeps the main copy of the document with a BPQ-value assigned (name, version). Thus, it can be retrieved by other nodes of a sharing group and cached within those as well as along the path. Any group node can make modifications to its local copy. The changes are propagated towards the origin node, from which the modified document is retrieved by the participating nodes.

**Access control:** To grant access to the document, we suggest using proxy signatures [1] to enable group members to apply modifications to content items stored in the network [11]. The origin distributes the proxy signing capability to the group members during an initial phase and signs those with its own private key. Caching nodes know or retrieve the corresponding public key securely (or it comes protected with the object). This enables them to validate modifications signed by authorized clients; they could also build a modification history that can be validated by others.

The proxy signing mechanism includes the steps to generate a warrant $\omega$ and the proxy certificate by the main server for all previously authenticated clients. Then the warrant and the certificate are distributed to the clients that makes them capable of signing the document. The cache can verify the signature based on the public keys of the main server and the client.

**Read and write:** Read operations utilize the BPQ requests and responses. The write operation is based on the new BPQ type value: `update`. When a new client requests a document from the cache, it generates a BPQ request with the initial name of the resource. The cache replies with the most recent version of the resource.

Then the client makes some modifications to the documents. It results in generating a patch file that includes the difference between the most recent cached version and the local client version. The client sends the patch in the BPQ "write" bundle, that includes the initial name of the document and the timestamp of the modifications. The cache applies the patch and replies to the client with the updated version of the document.

## 2.3 Preliminary Evaluation

Our initial evaluation of the framework implementation in a non-optimized system shows that the framework overhead itself —i.e., the hooks for the bundle content processing—is negligible. The overhead of searching the cache for available BPQ-resources grows with the number of bundles in the cache depending on the indexing data structures used. Measured in CPU cycles, the BPQ-aware node requires some 10% more time per bundle in case of cache miss with 1,000 bundles in the cache using a simple linear search. As for the document sharing application, where the bundles represent the mutations that will be rather small compared to the whole document and the number of parts will be about 1 to 10, the processing overhead is about 25%, scaling linearly with content size and modifications—but this obviously depends on the complexity of the operations.

# 3. CONCLUSION

In this paper, we take a first step towards mutable content in DTN/ICN environments and outline one application built on top of a general framework for content-based operations in DTNs. We deliberately focus on a simple scenario and leave parallel edits and conflict resolution in larger scale setups with multiple caches for future work. Yet, even the simple solution is able to preserve operations in subnets poorly connected to the Internet. The solution is feasible because the introduced overhead is minimal. Our future directions also include leveraging a more sophisticated key management schemes to provide Content-Based Information Security at diverse granularities, which gains importance for cooperative editing processes in larger groups.

# 4. REFERENCES

[1] A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25:57–115, 2012.

[2] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. RFC 4838, Apr. 2007.

[3] P. Costa, M. Musolesi, C. Mascolo, and G. P. Picco. Adaptive Content-based Routing for Delay-tolerant Mobile Ad Hoc Networks. Research Note 06-08, University College London, 2008.

[4] R. Krishnan et al. The SPINDLE Disruption-Tolerant Networking System. In *Proc. IEEE MILCOM*, 2007.

[5] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proc. of ACM SIGCOMM*, 2003.

[6] S. Farrell, A. Lynch, D. Kutscher, and A. Lindgren. Bundle protocol query extension block. Internet Draft draft-irtf-dtnrg-bpq-01, Work in progres, 2012.

[7] J. Ott. Delay Tolerance and the Future Internet. In *11th International Symposium on Wireless Personal Multimedia Communications*, 2008.

[8] M. Pitkänen, T. Kärkkäinen, J. Greifenberg, and J. Ott. Searching for Content in Mobile DTNs. In *Proc. of PerCom*, 3 2009.

[9] M. Pitkänen and J. Ott. Enabling Opportunistic Storage for Mobile DTNs. *Journal on Pervasive and Mobile Computing*, 4(5):579–594, October 2008.

[10] A. Sathiaseelan and J. Crowcroft. Lcd-net: lowest cost denominator networking. *SIGCOMM Comput. Commun. Rev.*, 43(2), April 2013.

[11] D. Schürmann, J. Ott, and L. Wolf. Authenticated resource mangement in delay-tolerant networks using proxy signatures. In *Proceedings of IEEE/IFIP WONS*, volume 1, 2013.

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