Workgroup: ICNRG
Internet-Draft:
draft-oran-icnrg-flowbalance-04
Published: 24 August 2020
Intended Status: Experimental
Expires: 25 February 2021
Authors: D. Oran
Network Systems Research and Design

Maintaining CCNx or NDN flow balance with highly variable data object sizes

Abstract

Deeply embedded in some ICN architectures, especially Named Data Networking (NDN) and Content-Centric Networking (CCNx) is the notion of flow balance. This captures the idea that there is a one-to-one correspondence between requests for data, carried in Interest messages, and the responses with the requested data object, carried in Data messages. This has a number of highly beneficial properties for flow and congestion control in networks, as well as some desirable security properties. For example, neither legitimate users nor attackers are able to inject large amounts of un-requested data into the network.

Existing congestion control approaches however have a difficult time dealing effectively with a widely varying MTU of ICN data messages, because the protocols allow a dynamic range of 1-64K bytes. Since Interest messages are used to allocate the reverse link bandwidth for returning Data, there is large uncertainty in how to allocate that bandwidth. Unfortunately, most current congestion control schemes in CCNx and NDN only count Interest messages and have no idea how much data is involved that could congest the inverse link. This document proposes a method to maintain flow balance by accommodating the wide dynamic range in Data message size.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Deeply embedded in some ICN architectures, especially Named Data Networking ([NDN]) and Content-Centric Networking (CCNx [RFC8569], [RFC8609]) is the notion of *flow balance*. This captures the idea that there is a one-to-one correspondence between requests for data, carried in Interest messages, and the responses with the requested data object, carried in Data messages. This has a number of highly beneficial properties for flow and congestion control in networks, as well as some desirable security properties. For example, neither legitimate users nor attackers are able to inject large amounts of un-requested data into the network.

This approach leads to a desire to make the size of the objects carried in Data messages small and near constant, because flow balance can then be kept using simple bookkeeping of how many Interest messages are outstanding. While simple, constraining Data messages to be quite small - usually on the order of a link Maximum Transmission Unit (MTU) - has some constraints and deleterious effects, among which are:

* Such small data objects are inconvenient for many applications; their natural data object sizes can be considerably larger than a link MTU.

* Applications with truly small data objects (e.g. voice packets in an Internet telephony applications) have no way to communicate that to the network, causing resources to still be allocated for MTU-sized data objects

* When chunking a larger data object into multiple Data messages, each message has to be individually cryptographically hashed and signed, increasing both computational overhead and overall message header size. The signature can be elided when Manifests are used (by signing the Manifest instead), but the overhead of hashing multiple small messages rather than fewer larger ones remains.

One approach which helps with the last of these is to employ fragmentation for Data messages larger than the Path MTU (PMTU). Such messages are carved into smaller pieces for transmission over the link(s). There are three flavors of fragmentation: end-to-end, hop-by-hop with reassembly at every hop, and hop-by-hop with cut-through of individual fragments. A number of ICN protocol architectures incorporate fragmentation and schemes have been proposed for both NDN and CCNx, for example in [Ghali2013]. Fragmentation alone does not ameliorate the flow balance problem however, since from a resource allocation standpoint both memory and link bandwidth must be set aside for maximum-sized data objects to avoid congestion collapse under overload.

The design space considered in this document does not however extend to arbitrarily large objects (e.g. 100's of kilobytes or larger). As the dynamic range of data object sizes gets very large, finding the right tradeoff between handling a large number of small data objects versus a single very large data object when allocating link and buffer resources becomes intractable. Further, the semantics of Interest-Data exchanges means that any error in the exchange results in a re-issue of an Interest for the entire Data object. Very large data objects represent a performance problem because the cost of retransmission when Interests are retransmitted (or re-issued) becomes unsustainably high. Therefore, the method we propose deals with a dynamic range of object sizes from very small (a fraction of a link MTU) to moderately large - about 64 kilobytes or equivalently about 40 Ethernet packets, and assumes an associated fragmentation scheme to handle link MTUs that cannot carry the Data message in a single link-layer packet.

The approach described in the rest of this document maintains flow balance under the conditions outlined above by allocating resources accurately based on expected Data message size, rather than employing simple interest counting.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Method to enhance congestion control with signaled size information in Interest Messages

Before diving into the specifics of the design, it is useful to consider how congestion control works in NDN/CCNx. Unlike the IP protocol family, which relies on end-to-end congestion control (e.g. TCP[RFC0793], DCCP[RFC4340], SCTP[RFC4960], QUIC[I-D.ietf-quic-transport]), CCNx and NDN employ hop-by-hop congestion control. There is per-Interest/Data state at every hop of the path and therefore for each outstanding Interest, bandwidth for data returning on the inverse path can be allocated. In many current designs, this allocation is done using simple Interest counting - by queueing and subsequently forwarding one Interest message from a downstream node, implicitly this provides a guarantee (either hard or soft) that there is sufficient bandwidth on the inverse direction of the link to send back one Data message. A number of congestion control schemes have been developed that operate in this fashion, for example [Wang2013],[Mahdian2016],[Song2018],[Carofiglio2012]. Other schemes, like [Schneider2016] neither count nor police interests, but instead monitor queues using AQM (active queue management) to mark or drop returning Data packets that have experienced congestion. It is worth noting that every congestion control algorithm has an explicit fairness goal and associated objective function (usually either [minmaxfairness] or [proportionalfairness]). If your fairness is to be based on resource usage, pure interest counting doesn't do the trick, since a consumer asking for large thing can saturate a link and shift loss to consumers asking for small things.

In order to deal with a larger dynamic range of Data message size, some means is required to allocate link bandwidth for Data messages in bytes with an upper bound larger than a Path MTU and a lower bound lower than a single link MTU. Since resources are allocated for returning Data based on arriving Interests, this information must be available in Interest messages.

Therefore, one key idea is the inclusion of an *expected data size* TLV in each Interest message. This allows each forwarder on the path taken by the Interest to more accurately allocate bandwidth on the inverse path for the returning Data message. Also, by including the expected data size, large objects will have a corresponding weight in resource allocation, maintaining link and forwarder buffering fairness. The simpler Interest counting scheme was nominally "fair" on a per-exchange basis within the variations of data that fit in a single PMTU packet because all Interests produced similar amounts of data in return. In the absence of such a field, it is not feasible to allow a large dynamic range in object size. While schemes like [Schneider2016] would not employ the expected data size to allocate reverse link bandwidth, they can still benefit from the information to affect the AQM congestion marking algorithm, preferentially marking data packets that exceed the expected data size from the corresponding Interest.

It is natural to ask whether the additional complexity introduced into an ICN forwarder, and the additional computational cost for the congestion control operations is worthwhile. For congestion control schemes like [Schneider2016] the additional overhead is not trivial, since no Interest counting is happening. However, if a congestion control is *already* counting Interests, the additional overhead is minimal, only reading one extra TLV from the Interest and incrementing the outstanding data amount for the corresponding queue by that number rather than a constant of 1. The overhead on returning data is simply reducing the amount by the actual Data message size, rather than by 1.

3.1. How to predict the size of returning Data messages

This of course raises the question "How does the requester know how big the corresponding Data message coming back will be?". For a number of important applications, the size is known a priori due to the characteristics of the application. Here are some examples:

- *For many sensor and other Internet-of-Things applications, the data is instrument readings which have fixed known size.

- *In video streaming, the data is output of a video encoder which produces variable sized frames. This information is typically made available ahead of time to the streaming clients in the form of a *Manifest* (e.g [DASH], FLIC [I-D.irtf-icnrg-flic]), which contains the names of the corresponding segments (or individual frames) of video and audio and their sizes.

*Internet telephony applications use vocoders that typically employ fixed-size audio frames. Therefore, their size is known either a priori, or via an initialization exchange at the start of an audio session.

The more complex cases arise where the data size is not known at the time the Interest must be sent. Much of the nuance of the proposed scheme is in how mismatches between the expected data size and the actual Data message returned are handled. The consumer can either under- or over-estimate the data size. In the former case, the under-estimate can lead to congestion and possible loss of data. In the latter case, bandwidth that could have been used by data objects requested by other consumers might be wasted. We first consider "honest" mis-estimates due to imperfect knowledge by the ICN application; later we consider malicious applications that are using the machinery to mount some form of attack. We also consider the effects of Interest aggregation if the aggregated Interests have differing expected data sizes. Also, it should be obvious that if the Data message arrives, the application learns its actual size, which may or may not be useful in adjusting the expected data size estimate for future Interests.

In all cases, the expected data size from the Interest can be incorporated in the corresponding Pending Interest Table (PIT) entry of each CCNx/NDN forwarder on the path and hence when a (possibly fragmented) Data object comes back, its total size is known and can be compared to the expected size in the PIT for a mismatch. Aside: In the case of fragmentation, we assume a fragmentation scheme in which the total Data message size can be known as soon as any one fragment is received (a reasonable assumption for most any well-designed fragmentation method, such as that in [Ghali2013]).

3.2. Handling 'too big' cases

If the returning Data message is larger than the expected data size, the extra data could result in either unfair bandwidth allocation or possibly data loss under congestion conditions. When this is detected, the forwarder has three choices:

1. It could forward the Data message anyway, which is safe under non-congestion conditions, but unfair and possibly unstable when the output link is congested

2. It could forward the data when un-congested (e.g. by assessing output queue depth) but drop it when congested

3. It could always drop the data, as a way of "punishing" the requester for the mis-estimate.

Either of the latter two strategies is acceptable from a congestion control point of view. However, it is not a good idea to simply drop the Data message with no feedback to the issuer of the Interest because the application has no way to learn the actual data size and retry. Further, recovery would be delayed until the failing Interest timed out. Therefore, an additional element needed in protocol semantics is the incorporation of a "Data too big" error message (achieved via the use of an "Interest Return" packet in CCNx).

Upon dropping data as above, the CCNx/NDN forwarder converts the normal Data message into an Interest Return packet containing the existing [RFC8609] T_MTU_TOO_LARGE error code and the actual size of the Data message instead of the original content. It propagates that back toward the client identically to how the original Data message would have been handled. Subsequent nodes upon receiving the T_MTU_TOO_LARGE error treat identically to other Interest Return errors. When the Interest Return eventually arrives back to the issuer of the Interest, the user MAY reissue the Interest with the correct expected data size.

One detail to note is that an Interest Return carrying T_MTU_TOO_LARGE must be deterministically smaller than the expected data size in all cases. This is clearly the case for large data objects, but there is a corner case with small data objects. There has to be a minimum expected data size that a client can specify in their Interests, and that minimum cannot be smaller than the size of a T_MTU_TOO_LARGE Interest Return packet.

3.3. Handling 'too small' cases

Next we consider the case where the returning data is smaller than the expected data size. While this case does not result in congestion, it can cause resources to be inefficiently allocated because not all of the set-aside bandwidth for the returning data object gets used. The simplest and most straightforward way to deal with this case is to essentially ignore it. The motivation for not worrying about the smaller data mismatch is that in many situations that employ usage-based resource measurement (and possibly charging), it is trivial to just account for the usage according to the larger expected data size rather than actual returned data size. Properly adjusting congestion control parameters to somehow penalize users for over-estimating their resource usage requires fairly heavyweight machinery, which in most cases is not warranted. If desired, any of the following mechanisms could be considered:

* Attempt to identify future Interests for the same object or closely related objects and allocate resources based on some retained state about the actual size of prior objects

* Police consumer behavior and decrease the expected data size in one or more future Interests to compensate

* For small objects, do more optimistic resource allocation on the links on the presumption that there will be some "slack" due to clients overestimating data object size.

3.4. Interactions with Interest Aggregation

One protocol detail of CCNx/NDN that needs to be dealt with is Interest Aggregation. Interest Aggregation, while a powerful feature for maintaining flow balance when multiple consumers send Interests for the same Named object, introduces subtle complications. Whenever a second or subsequent Interest arrives at a forwarder with an active PIT entry it is possible that those Interests carry different parameters, for example hop limit, payload, etc. It is therefore necessary to specify the exact behavior of the forwarder for each of the parameters that might differ. In the case of the expected data size parameter defined here, the value is associated with the ingress face on which the Interest creating the PIT entry arrived, as opposed to being global to the PIT entry as a whole. Interest aggregation interacts with expected data size if Interests from different clients contain different values of the expected data size. As above in Section 3.3, the simplest solution to this problem is to ignore it, as most error cases are benign. However, there is one problematic error case where one client provides an accurate expected data size, but another who issued the Interest first underestimates, causing both to receive a T_MTU_TOO_LARGE error. This introduces a denial of service vulnerability, which we discuss below together with the other malicious actor cases.

There are two cases to consider:

1. The arriving Interest carries an expected data size smaller than any of the values associated with the PIT entry.

2. The arriving Interest carries an expected data size larger than any of the values associated with the PIT entry.

For Case (1) the Interest can be safely aggregated since the upstream links will have sufficient bandwidth allocated based on the larger expected data size (assuming the original Interest's expected data size was itself sufficiently large to accommodate the actual size of the returning Data). On the other hand, should the incoming face have bandwidth allocated based on the larger existing Interest's expected data size, or on the smaller value in the arriving interest? Here there are two possible approaches:

a. Allocate based on the data size already in the PIT. In this case the consumer sending the earlier Interest can cause over-allocation of link bandwidth for other incoming faces, but there will not be a T_MTU_TOO_LARGE error generated for that Interest

b. Allocate based on the value in the arriving Interest. If the returning Data is in fact larger, generate a T_MTU_TOO_LARGE Interest Return on that ingress face, while successfully returning the Data message on any faces that do not exhibit a too small expected data size

It is RECOMMENDED that the second policy be followed. The reasons behind this recommendation are as follows:

1. The link can be congested quite quickly after the queuing decision is made, especially if the data has a long link-occupancy time, so this is a safer alternative.

2. The cost of returning the error is only one link RTT, since the consumer (or downstream forwarder) can immediately re-issue the Interest with the correct size and perhaps pick up the cached object from the upstream forwarder's Content Store.

3. Being optimistic and returning the data interacts with the behavior of aggregate resource control and resource accounting, which in turn raises the messy issue of whether to "charge" the consumer for the actual bandwidth used or only for the requested bandwidth in the expected data.

4. The rabbit hole goes deeper if you add differential QoS to the equation or consumers "playing games" and intentionally underestimating so their interests get satisfied when links aren't congested. This makes handling malicious actors (Section 4) more difficult.

For Case (2) above, the Interest MUST be forwarded rather than aggregated to prevent a consumer from mounting a denial of service attack by sending intentionally too small expected data size (see Section 4 for additional detail on this and other attacks). As above for Case (1) it is RECOMMENDED that policy (b) above be followed.

3.5. Operation when some Interests lack the expected data size option and some have it

Since the expected data size is an optional hop-by-hop packet field, forwarders need to be prepared to handle an arbitrary mix of packets containing or lacking this option. There are two general things to address.

First, we assume that any forwarder supporting expected data size is running a more sophisticated congestion control algorithm that one employing simple interest counting. The link bandwidth resource allocation is therefore based directly, or indirectly, on the expected Data size in bytes. Therefore, the forwarder has to assign a value to use in the resource allocation for the reverse link. This specification does not mandate any particular approach or a default value to use. However, in the absence on other guidance, it makes sense to do one of two things:

1. Pick a default based on the link MTU of the face on which the Interest arrived and use that for all Interests lacking an expected data size. This is likely to be most compatible with simple interest counting which would rate limit all incoming interests equally.

2. Configure some values for given Name prefixes that have known sizes. This may be appropriate for dedicated forwarders supporting single use cases, such as:

   * A forwarder handling IoT sensors sending very small Data messages

   * A forwarder handling real-time video with large average Data packets that exceed link MTU and are routinely fragmented

   * A forwarder doing voice trunking where the vocoders produce moderate sized packets, still much smaller than the link MTU

The second area to address is what to do if an interest lacking an expected Data size is responded to by a Data message whose size exceeds the default discussed above. It would be inappropriate to issue a T_MTU_TOO_LARGE error, since the consumer is unlikely to understand or deal correctly with that new error case. Instead, it is RECOMMENDED that the forwarder:

* Ignore the mismatch if the reverse link is not congested and return the requested Data message anyway.

* If the reverse link is congested, issue an Interest Return with the T_NO_RESOURCES error code

This specification does not define or recommend any particular algorithm for assessing the congestion state of the link(s) to carry the Data message downstream to the requesting consumers. It is assumed that a reasonable algorithm is in use, because otherwise even basic Interest counting forms of congestion control would not be effective.

4. Dealing with malicious actors

First we note that various known attacks in CCNx or NDN can also be mounted by users employing this method. Attacks that involve interest flooding, cache pollution, cache poisoning, etc. are neither worsened nor ameliorated by the introduction of the congestion control capabilities described here. However, there are two new vulnerabilities that need to be dealt with. These two new vulnerabilities involve intentional mis-estimation of data size.

The first is a consumer who intentionally over-estimates data size with the goal of preventing other users from using the bandwidth. This is at most a minor additional concern given the discussion of how to handle over-estimation by honest clients in Section 3.2. If one of the amelioration techniques described there is used, the case of malicious over-estimation is also dealt with adequately.

The second is a user who intentionally under-estimates the data size with the goal having its Interest processed while the other aggregated interests are not processed, thereby causing T_MTU_TOO_LARGE errors and denying service to the other users with overlapping requests. There are a number of possible mitigation techniques for this attack vector, ranging in complexity. We outline two below; there may be others as or more effective with acceptable complexity and overhead:

* (Simplest) A user sending Interests resulting in a T_MTU_TOO_LARGE error is treated similarly to users mounting interest flooding attacks; the a router aggregating Interests with differing expected data sizes rate limits the face(s) exhibiting these errors, thus decreasing the ability of a user to issue enough mis-estimated Interests to collide and generate Interest aggregation.

* An ICN forwarder aggregating Interests remembers in the PIT entry not only the expected data size of the Interest it forwarded, but the maximum of the expected data size of the other Interests it aggregated. If a T_MTU_TOO_LARGE error comes back, instead of propagating it, the forwarder MAY treat this as a transient error, drop the Interest Return, and re-forward the Interest using the maximum expected data size in the PIT (assuming it is is bigger). This recovers from the error, but the attacker can

still cause an extra round trip to the producer or to an upstream forwarder with a copy of the data in its Content Store.

5. Mapping to CCNx and NDN packet encodings

The only actual protocol needed is a TLV in Interest messages that states the size in bytes of the expected Data Message coming back, and in the Interest Return on a "too big" error to carry the actual data size. In the case of CCNx, this covers the encapsulated Data Object, but not the hop-by-hop headers.

5.1. Packet encoding for CCNx

For CCNx[RFC8569] there is a new hop-by-hop header TLV, and a new value of the Interest Return "Return Type".

Expected Data Size (for Interest messages), or Actual Data Size (for Interest Return messages) TLV

Abbrev	Name	Description
T_DATASIZE	Data Size	Expected (Section 3) or Actual (Section 3.2) Data Size

Table 1: Data Size TLV

5.2. Packet encoding for NDN

TBD based on [NDNTLV]. Suggestions from the NDN team greatly appreciated.

6. IANA Considerations

Please Add the T_DATASIZE TLV to the Hop-by-Hop TLV types registry of RFC8609, with fixed length of 2, and data type numeric

Expected/Actual Data Size TLV encoding. The range has an upper bound of 64K bytes, since that is the largest MTU supported by CCNx.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|             T_DATASIZE        |               2               |
+---------------+---------------+---------------+---------------+
|   Expected/Actual Data Size   |
+---------------+---------------+
```

Figure 1: Expected/Actual Datazize using RFC8609 encoding

7. Security Considerations

Section 4 addresses the major security considerations for this specification.

8. Acknowledgements

Klaus Schneider and Ken Calvert have contributed a number of useful comments which have substantially improved the document.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

9.2. Informative References

[Carofiglio2012] Carofiglio, G., Gallo, M., and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks, in ICN Workshop at SIGcomm 2012", DOI 10.1145/2377677.2377772, 2102, <http://conferences.sigcomm.org/sigcomm/2012/paper/icn/p37.pdf>.

[DASH] "Dynamic Adaptive Streaming over HTTP", various, <https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP>.

[Ghali2013] Ghali, C., Narayanan, A., Oran, D., Tsudik, G., and C. Wood, "Secure Fragmentation for Content-Centric Networks, in IEEE 14th International Symposium on Network Computing and Applications", DOI 10.1109/nca.2015.34, 2015, <http://dx.doi.org/10.1109/NCA.2015.34>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-27, 21 February 2020, <https://tools.ietf.org/html/draft-ietf-quic-transport-27>.

[I-D.irtf-icnrg-flic]
Tschudin, C., Wood, C., Mosko, M., and D. Oran, "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-02, 4 November 2019, <https://tools.ietf.org/html/draft-irtf-icnrg-flic-02>.

[Mahdian2016] Mahdian, M., Arianfar, S., Gibson, J., and D. Oran, "MIRCC: Multipath-aware ICN Rate-based Congestion Control, in Proceedings of the 3rd ACM Conference on Information-Centric Networking", DOI 10.1145/2984356.2984365, 2016, <http://conferences2.sigcomm.org/acm-icn/2016/proceedings/p1-mahdian.pdf>.

[minmaxfairness] "Max-min Fairness", various, <https://en.wikipedia.org/wiki/Max-min_fairness>.

[NDN] "Named Data Networking", various, <https://named-data.net/project/execsummary/>.

[NDNTLV] "NDN Packet Format Specification.", 2016, <http://named-data.net/doc/ndn-tlv/>.

[proportionalfairness] "Proportionally Fair", various, <https://en.wikipedia.org/wiki/Proportionally_fair>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI

10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[Schneider2016] Schneider, K., Yi, C., Zhang, B., and L. Zhang, "A Practical Congestion Control Scheme for Named Data Networking, in Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16", DOI 10.1145/2984356.2984369, 2016, <http://conferences2.sigcomm.org/acm-icn/2016/proceedings/p21-schneider.pdf>.

[Song2018] Song, J., Lee, M., and T. Kwon, "SMIC: Subflow-level Multi-path Interest Control for Information Centric Networking, in 5th ACM Conference on Information-Centric Networking", DOI 10.1145/3267955.3267971, 2018, <https://conferences.sigcomm.org/acm-icn/2018/proceedings/icn18-final62.pdf>.

[Wang2013] Wang, Y., Rozhnova, N., Narayanan, A., Oran, D., and I. Rhee, "An Improved Hop-by-hop Interest Shaper for Congestion Control in Named Data Networking, in ACM SIGCOMM Workshop on Information-Centric Networking", DOI 10.1145/2534169.2491233, 2013, <http://conferences.sigcomm.org/sigcomm/2013/papers/icn/p55.pdf>.

Author's Address

Dave Oran
Network Systems Research and Design
4 Shady Hill Square
Cambridge, MA 02138
United States of America

Email: daveoran@orandom.net