Network Working Group L. Eggert
Internet-Draft Nokia
Intended status: Experimental January 27, 2011
Expires: July 31, 2011

Congestion Control for the Constrained Application Protocol (CoAP)
draft-eggert-core-congestion-control-01

Abstract

The Constrained Application Protocol (CoAP) is a simple, low-overhead, UDP-based protocol for use with resource-constrained IP networks and nodes. CoAP defines a simple technique to individually retransmit lost messages, but has no other congestion control mechanisms.

This document motivates the need for additional congestion control mechanisms, and defines some simple strawman proposals. The goal is to encourage experimentation with these and other proposals, in order to determine which mechanisms are feasible to implement on resource-constrained nodes and are effective in real deployments.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 31, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

The Constrained Application Protocol (CoAP) [I-D.ietf-core-coap] is a simple, low-overhead, UDP-based protocol for use with resource-constrained IP networks and nodes.

CoAP defines two kinds of interactions between end-points:

1. a client/server interaction model, where request or notify messages initiate a transaction with a server, which may send a response to the client with a matching transaction ID

2. an asynchronous subscribe/notify interaction model, where a server can send notify messages to a client about a resource which the client has subscribed to

CoAP uses the User Datagram Protocol (UDP) [RFC0768] to transmit these messages. For reliable messages, i.e., messages for which a delivery confirmation is required, CoAP defines a simple mechanism to individually retransmit such "confirmable" messages for which no delivery acknowledgement was received. This mechanism uses an exponentially backed-off timer to schedule a fixed number of re-transmission attempts.

This document argues that although this retransmission mechanism is a required first step to implement congestion control for CoAP, it alone is not sufficient to alleviate network overload in all conditions. Section 2 gives a short summary of Internet congestion control principles, and Section 3 presents some simple strawman proposals that attempt to complement the current message retransmission mechanism in CoAP.

2. Discussion of Internet Congestion Control Principles

[RFC2914] describes the best current practices for congestion control in the Internet, and requires that Internet communication employ

congestion control mechanisms. Because UDP itself provides no congestion control mechanisms, it is up to the applications and application-layer protocols that use UDP for Internet communication to employ suitable mechanisms to prevent congestion collapse and establish a degree of fairness. CoAP is one such application-layer protocol.

[RFC2914] identifies two major reasons why congestion control mechanisms are critical for the stable operation of the Internet:

1. The prevention of congestion collapse, i.e., a state where an increase in network load results in a decrease in useful work done by the network.

2. The establishment of a degree of fairness, i.e., allowing multiple flows to share the capacity of a path reasonably equitably.

Bulk transfers cause the overwhelming majority of the bytes on the Internet, and the traditional congestion control mechanisms used for bulk transfers are engineered to saturate the network without driving it into congestive collapse. Fairness between flows is an important secondary consideration when the network operates around the saturation point, so that new flows are not disadvantaged compared to established flows, and can obtain a reasonable share of the capacity quickly.

The environments that CoAP targets are IP networks, although more resource-constrained ones than the "big-I" Internet. This does not eliminate the need for end-point-based congestion control! If anything, the environments that CoAP will be deployed in have fewer capabilities for network provisioning, queuing and queue management, traffic engineering and capacity allocation, which are among the techniques that can sometimes offset the need for end-to-end congestion control to some degree.

However, the environments that CoAP targets are sufficiently different from the "big-I" Internet so that the motivations for congestion control from [RFC2914] should probably be weighted differently. CoAP networks will not be used for bulk data transfers and CoAP nodes will not need to use a significant fraction of the capacity of a path to provide a useful service. (In fact, they are often too resource-constrained to do so in the first place.) Under normal operation, a CoAP network will be mostly idle, which means that fairness between the transmissions of different CoAP nodes is not a large issue. A CoAP congestion control mechanism can hence focus on preventing congestion collapse, i.e., preventing situations where the mount of useful work done approaches zero as network load

increases. This is a much more tractable problem given the specific conditions of CoAP environments.

The current IETF congestion control mechanisms, such as TCP [RFC5681] or TFRC [RFC5348], all focus on determining a "safe" sending rate for a bulk transfer, i.e., for a single flow of many packets between a sender and destination where many packets are in flight at any given time. They measure the path characteristics, such as round-trip time (RTT) and packet loss rate, by monitoring the ongoing transfer and use this information to adjust the sending rate of the flow during the transmission.

This approach is not feasible for CoAP. The infrequent request/response interaction that CoAP supports does not generate sufficient data about the path characteristics to drive a traditional congestion control loop, even if the notion of "a flow" to a destination is extended from "one CoAP transaction" to "a sequence of CoAP transactions". Further complications can arise for CoAP deployments that involve low-capacity, low-power radio links that can cause highly variable path characteristics that are more challenging to adapt to than traditional "big-I" Internet paths. This approach is also not applicable to multicast transmissions, which may see frequent use in some CoAP deployments.

[RFC5405] documents the IETF's current best practices for using UDP for unicast communication in the Internet. It provides guidance on topics such as message sizes, reliability, checksums, middlebox traversal and congestion control. Section 3.1.2 of [RFC5405], which focuses on congestion control for low data-volume applications, is especially relevant to CoAP.

Section 3.1.2 of [RFC5405] acknowledges that the traditional IETF congestion control mechanisms are not applicable for low data-volume application protocols such as CoAP. Instead, it recommends that such application protocols:

- o maintain an estimate of the RTT for any destination with which they communicate, or assume a conservative fixed value of 3 seconds when no RTT estimate can be obtained (e.g., unidirectional communication)

- o control their transmission behavior by not sending on average more than one UDP datagram per RTT to a destination

- o detect packet loss and exponentially back their retransmission timer off when a loss event occurs

- o employ congestion control for both directions of a bi-directional communication

CoAP follows some of these guidelines already. At the moment, it uses a fixed value of 2 seconds for its retransmission timer for both requests and responses, which although somewhat shorter than the recommended value in [RFC5405] is likely appropriate for many of its deployment scenarios. CoAP also uses exponential back-off for its retransmission timer.

This alone, however, does not result in a complete congestion control mechanism for CoAP. Section 3 defines an experimental complement to the current CoAP mechanism described in [I-D.ietf-core-coap].

3. CoAP Congestion Control

This section proposes several congestion control techniques for CoAP that are intended to improve its ability to prevent congestion collapse. At the moment, these techniques are described with the intent of encouraging experimentation with such proposals in CoAP simulations and experimental testbed deployments. Of particular interest are mechanism requiring little computation and state, i.e., mechanisms that can be implemented in resource-constrained nodes without much overhead.

3.1. Retransmissions

CoAP already defines a simple retransmission scheme with exponential back-off, where messages that have not been responded to in RESPONSE_TIMEOUT are retransmitted, followed by doubling RESPONSE_TIMEOUT. Up to MAX_RETRANSMIT retransmission attempts are made. (At the moment, [I-D.ietf-core-coap] defines RESPONSE_TIMEOUT to be 2 seconds and MAX_RETRANSMIT to be four attempts.) As stated above, although RESPONSE_TIMEOUT is somewhat shorter than what [RFC5405] recommends, the shorter value is likely to not cause large issues in many deployments that CoAP targets.

However, using a fixed value for RESPONSE_TIMEOUT instead of basing it on the measured RTT to a destination has some minor drawbacks. CoAP may be used in deployments where the path RTTs can approach the currently defined RESPONSE_TIMEOUT of 2 seconds, such as Internet deployments involving GSM or 3G links, or cases where preparing a response can involve significant computation or where it otherwise incurs delays, such as long sleep cycles at the receiver. Fixed timeouts that are too short can cause spurious retransmissions, i.e., unnecessary retransmissions in cases where either the request or the response are still in transit. Spurious retransmissions, especially

persistent ones, waste resources.

This section therefore proposes that CoAP deployments experiment with maintaining an estimate of the RTT for any destination with which they (frequently) communicate. Specifically, it is suggested that deployments experiment with the algorithm specified in [RFC2988] to compute a smoothed RTT (SRTT) estimate, and compute RESPONSE_TIMEOUT in the same way [RFC2988] computes RTO.

This suggestion unfortunately does require maintaining per-destination state at the sender, which may be undesirable. The amount of required state can be reduced by maintaining a single "upper bound" RTT measurement across all destinations. The downside here is that retransmissions may be delayed longer than they would be with per-destination state; the upside is that multicast messages are supported.

A second suggestion is to experiment with a longer RESPONSE_TIMEOUT, such as 3 seconds or longer, which is what [RFC5405] recommends, in order to determine if there are significant drawbacks or whether this default value could be lengthened.

3.2. Aggregate Congestion Control

Traditional Internet congestion control algorithms control the sending rate of a single flow. When a node establishes multiple, parallel flows, their congestion control loops run (mostly) independently of one another. Interactions between the control loops of parallel flows are (mostly) indirect, e.g., a rate increase of one flow may cause packet loss and an eventual rate decrease to another.

CoAP "flows", i.e., sequences of infrequent CoAP transactions between the same two nodes, do not require much more per-flow congestion control than a retransmission scheme that reduces the rate (increases the back-off) of a flow under loss, and a (low) cap on the number of allowed outstanding requests to a destination. ([RFC5405] recommends "on average not more than one" outstanding transaction to a given destination.)

On the other hand, CoAP applications may potentially want to initiate many transactions with different nodes at the same time. Allowing CoAP applications to initiate an unlimited number of parallel transactions gives them the means for causing overload, and depends on application-level measures to detect and correctly mitigate this failure. Because each transaction only consumes a very limited amount of resources, it is arguably more important to control the total outstanding number of transactions, compared to controlling the rate at which each individual one is being (re)transmitted. The CoAP

spec [I-D.ietf-core-coap] does currently not impose any limit on how many parallel transactions to different nodes an end-point may have outstanding.

Given the importance of preventing congestion collapse, this document argues that the CoAP protocol should specify a common mechanism for congestion controlling the aggregate traffic a CoAP node sends into the network. In other words, the CoAP stack should locally drop application-generated messages under overload situations (or indicate to applications that at the moment, no transmission is permissible), rather than attempting to send them into the network, irrespective of the destination.

One proposal is to implement a simple windowing algorithm. In this mechanism, a CoAP node has a certain number of "transmission credits" available during a time interval. Sending one CoAP message consumes one transmission credit, independent of which destination it is being sent to. If all transmission credits have been used up during a time interval, the CoAP node drops any additional messages that the applications attempt to send during the remainder of the time interval (or it prevents applications from generating the messages in the first place). At the end of a time interval, the CoAP node determines whether acknowledgments have been received for all "confirmable" messages it has sent within the time interval. If this is the case, the CoAP node increases the number of transmission credits by one for the following time interval. If acknowledgments fail to arrive for some of the "confirmable" messages sent during the time interval, the number of transmission credits is cut in half for the next interval.

The description above leaves several questions unanswered. These include the length of the time interval and whether it is fixed or adapted over time, whether an increase by one and a reduction by half are the correct parameters for the proposed AIMD (additive increase, multiplicative decrease) scheme, whether the decrease should be proportional to the loss rate, how non-confirmable and multicast messages are handled, and others.

At the moment, this document does not attempt to answer these questions. Instead, it encourages simulations and implementations to explore the design space, and also consider other non-windowing approaches.

3.3. Explicit Congestion Notification

Explicit Congestion Notification (ECN) [RFC3168] is an extension to IP that allows routers to inform end nodes when they approach congestion by setting a bit in the IP header. The receiver of a

message echoes this bit to the sender, which reacts as if packet loss had occurred for the flow.

Deployment of ECN can reduce overall packet loss, because senders can react to congestion early, i.e., before packet loss occurs. This is especially attractive in resource-constrained environments, because retransmissions can be avoided, which conserves resources.

If CoAP uses an aggregate congestion control mechanism such as described in Section 3.2, it will reduce the amount of transmission credits for the next time interval when some of the responses received had the ECN bit set. (Other reactions to ECN markings may be possible.)

Whether ECN support is possible in CoAP deployments remains to be investigated, because ECN usage requires a negotiation handshake (can potentially be avoided if support is made mandatory for CoAP deployments) and because routers need to support ECN marking. At this point, simulations attempting to quantify the benefits may therefore be easiest to obtain in order to understand which benefits ECN brings to CoAP.

3.4. Multicast Considerations

CoAP requests may be multicast, and result in several replies from different end-points, potentially consuming much more resource capacity for the request and response transmissions than a single unicast transaction. It can therefore be argued that the sending multicast requests should be more conservatively controlled than the sending of unicast requests.

CoAP already acknowledges this to some degree by not retransmitting multicast requests at the CoAP level. Unfortunately, CoAP currently has no means for preventing an application from doing application-level retransmissions of multicast requests. Given that the prevention of congestion collapse is important, such a mechanism should be added.

The aggregate congestion control proposal in Section 3.2 puts a cap on the number of transmissions allowed during a time interval, including multicast requests. It is currently unclear whether additional means are required for CoAP deployments that make heavy use of multicast. As before, experimentation is encouraged to understand the problem space.

4. IANA Considerations

This document requests no actions from IANA.

[Note to the RFC Editor: Please remove this section upon publication.]

5. Security Considerations

This document has no known security implications.

[Note to the RFC Editor: Please remove this section upon publication.]

6. Acknowledgments

Lars Eggert is partly funded by [TRILOGY], a research project supported by the European Commission under its Seventh Framework Program.

7. References

7.1. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-04 (work in progress), January 2011.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

7.2. Informative References

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[TRILOGY] "Trilogy Project", http://www.trilogy-project.org/.

Author's Address

Lars Eggert
Nokia Research Center
P.O. Box 407
Nokia Group 00045
Finland

Phone: +358 50 48 24461
Email: lars.eggert@nokia.com
URI: http://research.nokia.com/people/lars_eggert