TCP Parameter Dynamic Control
draft-song-dclc-tcpdc-00

Abstract

This document describes a framework and message flows for centralized TCP parameter control, so that each end host in a network can make better use of the network resource according to the network status. A TCP Optimization Element and a TCP Optimization Agent are introduced. The message patterns include request response and subscription/notification. This mechanism can be used in network service providers’ networks, as well as in data center networks.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2014 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.

Table of Contents

1 Introduction .................................................. 3
2 Conventions Used in This Document ......................... 4
3 TCP Parameter Control Architecture ........................ 4
  3.1 Guidance Level ........................................... 5
  3.2 Subscription Mode ........................................ 6
  3.3 Request/Response Mode .................................... 6
4 Messages ...................................................... 6
  4.1 Explicit RR ................................................ 6
    4.1.1 TcpParReq ............................................ 6
    4.1.2 TcpParRes ............................................ 7
  4.2 Subscription/Notification .................................. 7
    4.2.1 TcpParSub ............................................ 7
    4.2.2 Notification ......................................... 8
5 Security Considerations ....................................... 9
6 Acknowledgement ............................................... 9
7 IANA Considerations ......................................... 9
8 References .................................................... 9
  8.1 Normative References ..................................... 9
Authors’ Addresses ............................................. 9
1 Introduction

Here is a figure (Figure 1) indicating utilization rate of a very busy backbone link. We can see that even for this busy link, there are average 6 to 8 hours when the utilization rate is below 50% for each day. And if the TCP timeout happens during this period, the sender will also reduce the congestion window, for example to half size. This is actually not necessary. Because timeout is caused by packet loss, but packet loss does not necessarily mean network link congestion. However, the sender of this TCP connection does not know the network status. When the TCP was designed, it was a very useful way to avoid congestion. But time has changed, and now there are many ways to dynamically or statistically monitor the network status, so it becomes possible to notify the application endpoints and make them utilize the network resources in a more efficient way.

The forwarding capacity of the network is evolving very fast nowadays. When the TCP was designed, the routers and switches have low capacity, and the network was easy to be congested. So it was designed with a very small initial congestion window. But small initial congestion window size means more cycles during the slow start period. So for Linux 3.0, Google proposed to increase the init_cwnd. For example, when 1095 < MSS <= 2190, the original init_cwnd = 3, but in Linux 3.0, Google proposes to increase it to 10. However, that’s still a fixed number without considerations of
the network variations. In some areas of the world, the network condition is much better than that of other areas. That init_cwnd size should be even bigger to provide better performance for applications inside that area (when both sender and receiver are inside that area).

So the basic suggestion is to allow end hosts to dynamically adjust its TCP parameters according to the network status, with additional consideration of location of both endpoints, time of a day and endpoint status. For example, during the period when the network utilization is under 50%, the end hosts can adjust their init_cwnd to 20 segments, and keep the init_cwnd unchanged when the network utilization rate is between 50% and 60%. But when the network utilization is above 60%, the end hosts can adjust their init_cwnd to 10 again. And for another example, when an end host in a high speed network area communicates with another end host in the same area, it is also possible to adjust the init_cwnd to 20, but when it communicates with another end host in a bad network condition area, the init_cwnd should be set a small value.

It is also possible to change the TCP timeout behaviors according to the network status. When the timeout happens during the period that relative network link utilization is under 50% (the cwnd size does not exceed the peak buffer size, and the rate does not exceed the subscription rate), the cwnd can be remained the same, without reducing it tremendously, if the sending rate does not exceed the subscription rate (upload rate of the sender and download rate of the receiver) nor overflow the receiver's receiving window.

2 Conventions Used in This Document

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 [KEYWORDS]. This document also uses the following conventions.

TOE: TCP Optimization Element, which accesses the network statistical information from network measurement entities, such as an OAM server, NMS, or a LMAP server and etc, and provides the TCP optimization service to the TCP Optimization Agent (TOA).

TOA: TCP Optimization Agent, which is deployed in the end host, and adjust the TCP stack behavior according to the guidance from the TOE. Note that one TOA can serve multiple applications.

3 TCP Parameter Control Architecture
It is assumed that there is an existing method for the TOE to get the routing information and network status for each link in a network, for example, from a PCE server. Then the TOE knows the possible path for each communication, and it also knows about the link utilization rate, lost ratio, and the statistics information of the link and the network. The TOE contemplates the network utilization rate at different time during a day, and sets the TCP optimization parameters accordingly. For example, from the midnight to early morning, the network utilization is very low, end hosts can use larger init_cwnd size and the window size degradation behavior can be much slower during time-out or receiving the same ACK event.

3.1 Guidance Level

There are different types of guidance from the TOE according to different network levels.

The normal type would be the TCP optimization parameter for the whole administrative network domain. When source end host and the destination end host are inside the same administrative network domain, they are suggested to use the parameters provided by the TOE to optimize the TCP transport. The domain can be an intra DC network, a LAN network or a NSP network.

Another type is TCP optimization parameter for a particular link, for example, TOE provides optimization parameters to end hosts in two data centers which share an inter-DC dedicated link. When the link is congested, the TOE suggests the end hosts to use smaller init_cwnd
size and reduce the congestion window sharply during time-out or replicated ACKs. This type of service is only available when the source end host and the destination end host are deployed at two ends of a particular link.

When either one of the communication endpoints is out of the scope of the administrative boundaries, the recommendation TCP optimization parameters MUST NOT be used.

### 3.2 Subscription Mode

TOA can use subscription mode to communicate with the TOE to get updated TCP optimization parameters. This is very useful for long-lived traffic, as well as for end hosts which have frequent TCP connections. The guidance level can be either the network level or the link level.

### 3.3 Request/Response Mode

TOA can also use the request response mode to communicate with the TOE. With each TCP optimization request, the TOA lists the two communication end hosts IP address, and indicate the level of guidance. Then TOE gives the response of the current recommendation parameters for TCP transport.

### 4 Messages

A TOA uses the HTTP protocol with an HTTP POST entity body of JSON Objects, to request the TCP parameter guidance from a TOE server.

#### 4.1 Explicit RR

Explicit request and response mode is mainly used for the guidance of TCP parameters between two endpoints. If the path between two endpoints is a dedicated link, it is easier to give the guidance with considering the two endpoint properties and the link utilization status. When the path between two endpoints is within the administrative domain of the TOE, but subject to change (for example, the route may be changed through routers), then the TOE should give conservative guidance parameters.

#### 4.1.1 TcpParReq

```json
object {
    TypedEndpointAddr: source;
    TypedEndpointAddr: destination;
} TcpParReq;
```
Typed Endpoint Address: Typed Endpoint Addresses are encoded as strings of the format ‘AddressType:EndpointAddr’, with the ‘:’ character as a separator. The type ‘TypedEndpointAddr’ is used to indicate a string of this format. This document defines two values for AddressType: ‘ipv4’ to refer to IPv4 addresses, and ‘ipv6’ to refer to IPv6 addresses. EndpointAddr component of TypedEndPointAddr is also encoded as a string. The exact characters and format depend on AddressType. This document defines EndpointAddr when AddressType is ‘ipv4’ or ‘ipv6’. IPv4 Endpoint Addresses are encoded as specified by the ‘IPv4address’ rule in Section 3.2.2 of [RFC3986]. IPv6 Endpoint Addresses are encoded as specified in Section 4 of [RFC5952].

Upon receive this request, TOE should lookup the subscription rate, i.e. uplink rate quota of the source and the downlink rate quota of the destination, and then examine the current link utilization rate, then gives the appropriate TCP parameter guidance.

The media type for explicit request is "application/tcpdc-rr+json".

4.1.2 TcpParRes

    object {
        TcpPar: parameters<0...*>;
    } TcpParRes;

    object {
        ParType -> ParValue;
    } TcpPar;

ParType: A JSONString defined the TCP parameter type, this document defines the "initcwnd", "threshold", "timeOut", and "repeatedtimeouts". (It is open for discussion).

ParValue: A JSONValue defined the value for the relative parameter type.

The media type for explicit response is "application/tcpdc-rrparameters+json".

4.2 Subscription/Notification

This method is mainly used for getting the guidance for the TCP parameters in the administrative domain, but can also be used for long-lived traffic flows. In the response, it has indications on when to change the TCP parameters.

4.2.1 TcpParSub
object {
    TypedEndpointAddr: source;
    [TypedEndpointAddr: destination;]
    GuidanceLevel: level;
}TcpParSub

TypedEndpointAddr: the same as defined in previous sections.

GuidanceLevel: A JSONString which defines the level of guidance. This document defines the value of "link" and "AS".

Destination address is optional. When the source end host sends subscription for its TCP parameter guidance on the administrative domain, it does not need the destination address. However, when the end host sends subscription for the link, it has to provide the destination address.

The media type for subscription is "application/tcpdc-sub+json".

Sending subscription message to the TOE without any json object in the message body means unsubscription.

4.2.2 Notification

object {
    ConditionedTcpPar: cparameters<0...*>;  
}TcpParNotify

object {
    Condition conditions<0...*>;
    TcpPar: parameters<0...*>;
}ConditionedTcpPar;

Condition: A condition contains three entities separated by whitespace: (1) a JSONString indicated the link or network status, or the subscriber property, this document defines "link-utilization-rate", "network-utilization-rate", "source-uplink-sub-rate", and "destination-download-sub-rate". (2) an operator, 'gt' for greater than, 'lt' for less than, 'ge' for greater than or equal to, 'le' for less than or equal to, or 'eq' for equal to; (3) a target JSONValue. The JSONValue is a number indicated to compare with the previous status.

The media type for subscription is "application/tcpdc-notify+json".

The TCP parameter guidance will be sent to the IP address/port which subscribed earlier. When the template has changed, the TOE will send an immediate notification to relative TOAs.
Note that the guidance delivers the message such as when network utilization is between 50% to 80%, then the recommended parameters are given. So it means the TOA also has to get the change of the relative network status. Network or link status notification was assumed to be provided by other protocols, but if needed, this document can also be expanded to deliver the relative status. (Open issue)

5 Security Considerations

Dynamic control of TCP parameters can be used for attacks and can cause serious problems to the network or to the applications.

If there are no proper mechanisms to monitor the network, it may be used to maliciously change the TCP parameters and cause network congestion. But in most environments it can be avoided as there are rate limitations.

It can also be used to attack the end hosts. So a mechanism to protect the illegal modification is needed.

6 Acknowledgement

Lingli Deng has provided many valuable comments to this document.

7 IANA Considerations

TBD.

8 References

8.1 Normative References


Authors’ Addresses