TCPM M. Scharf
Internet-Draft Hochschule Esslingen
Intended status: Standards Track V. Murgai
Expires: August 27, 2020 Cisco Systems Inc
M. Jethanandani
VMware
February 24, 2020

YANG Model for Transmission Control Protocol (TCP) Configuration
draft-scharf-tcpm-yang-tcp-04

Abstract

This document specifies a YANG model for TCP on devices that are configured by network management protocols. The YANG model defines a container for all TCP connections and groupings of fundamental parameters that can be imported and used in many TCP implementations. The model includes definitions from YANG Groupings for TCP Client and TCP Servers (I-D.ietf-netconf-tcp-client-server). The model is NMDA (RFC 8342) compliant.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Transmission Control Protocol (TCP) [RFC0793] is used by many applications in the Internet, including control and management protocols. Therefore, TCP is implemented on network elements that can be configured via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. This document specifies a YANG [RFC7950] 1.1 model for configuring TCP on network elements that support YANG data models, and is Network Management Datastore Architecture (NMDA) [RFC8342] compliant. This module defines a container for TCP connection, and includes definitions from YANG Groupings for TCP Clients and TCP Servers [I-D.ietf-netconf-tcp-client-server]. The model focuses on fundamental and standard TCP functions that are widely implemented. The model can be augmented to address more advanced or implementation-specific TCP features.

Many protocol stacks on Internet hosts use other methods to configure TCP, such as operating system configuration or policies. Many TCP/IP stacks cannot be configured by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. Moreover, many existing TCP/IP stacks do not use YANG data models. Such TCP implementations

often have other means to configure the parameters listed in this document, which are outside the scope of this document.

This specification is orthogonal to Management Information Base (MIB) for the Transmission Control Protocol (TCP) [RFC4022]. The TCP Extended Statistics MIB [RFC4898] is also available. It is possible to translate a MIB into a YANG model, for instance using Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules [RFC6643]. However, this approach is not used in this document, as such a translated model would not be up-to-date.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Model Overview

3.1. Modeling Scope

TCP is implemented on many different system architectures. As a result, there are may different and often implementation-specific ways to configure parameters of the TCP protocol engine. In addition, in many TCP/IP stacks configuration exists for different scopes:

- o Global configuration: Many TCP implementations have configuration parameters that affect all TCP connections. Typical examples include enabling or disabling optional protocol features.

- o Interface configuration: It can be useful to use different TCP parameters on different interfaces, e.g., different device ports or IP interfaces. In that case, TCP parameters can be part of the interface configuration. Typical examples are the Maximum Segment Size (MSS) or configuration related to hardware offloading.

- o Connection parameters: Many implementations have means to influence the behavior of each TCP connection, e.g., on the programming interface used by applications. A typical example are socket options in the socket API, such as disabling the Nagle algorithm by TCP_NODELAY. If an application uses such an interface, it is possible that the configuration of the application or application protocol includes TCP-related parameters. An example is the BGP YANG Model for Service Provider Networks [I-D.ietf-idr-bgp-model].

o Policies: Setting of TCP parameters can also be part of system policies, templates, or profiles. An example would be the preferences defined in the TAPS interface An Abstract Application Layer Interface to Transport Services [I-D.ietf-taps-interface].

As a result, there is no ground truth for setting certain TCP parameters, and traditionally different TCP implementation have used different modeling approaches. For instance, one implementation may define a given configuration parameter globally, while another one uses per-interface settings, and both approaches work well for the corresponding use cases. Also, different systems may use different default values.

In addition, TCP can be implemented in different ways and design choices by the protocol engine often affect configuration options. In a number of areas there are known differences between different TCP stack architectures. This includes amongst others:

o Window size: TCP stacks can either store window state variables (such as the congestion window) in segments or in bytes.

o Buffer sizes: The memory management depends on the operating system. As the size of buffers can vary over several orders of magnitude, very different implementations exist. This typically influences TCP flow control.

o Timers: Timer implementation is another area in which TCP stacks may differ.

Nonetheless, there are a number of basic system parameters that are configurable on many TCP implementations, even if not all TCP implementations may indeed have all these settings, and even if there are differences regarding syntax and semantics. This document focuses on modeling such basic parameters directly following from standards.

In addition to configuration of the TCP protocol engine, a TCP implementation typically also offers access to operational state and statistics. This includes amongst others:

o Statistics: Counters for the number of active/passive opens, sent and received segments, errors, and possibly other detailed debugging information

o TCP connection table: Access to status information for all TCP connections

o TCP listener table: Information about all TCP listening endpoints

3.2. Model Design

This document models fundamental system parameters that are configurable on many TCP implementations, and for which the configuration is reasonably similar. Similar to the TCP MIB [RFC4022], this document also specifies a TCP connection table. This enables both global and connection-specific TCP configuration.

An important use case is the TCP configuration on network elements such as routers, which often use YANG data models. The model therefore specifies TCP parameters that are important on such TCP stacks. A typical example is the support of TCP-AO [RFC5925]. TCP-AO is increasingly supported on routers and it requires configuration.

The YANG model defined in this document includes definitions from the YANG Groupings for TCP Clients and TCP Servers [I-D.ietf-netconf-tcp-client-server]. Similar to that model, this specification defines YANG groupings. This allows reuse of these groupings in different YANG data models. It is intended that these groupings will be used either standalone or for TCP-based protocols as part of a stack of protocol-specific configuration models. An example could be the BGP YANG Model for Service Provider Networks [I-D.ietf-idr-bgp-model].

There are also other existing TCP-related YANG models, which are othogonal to this specification. Examples are:

- TCP header attributes are modeled in other models, such as YANG Data Model for Network Access Control Lists (ACLs) [RFC8519] and Distributed Denial-of-Service Open Thread Signaling (DOTS) Data Channel Specification [I-D.ietf-dots-data-channel].

- TCP-related configuration of a NAT (e.g., NAT44, NAT64, Destination NAT, ...) is defined in A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT) [RFC8512] and A YANG Data Model for Dual-Stack Lite (DS-Lite) [RFC8513].

3.3. Tree Diagram

This section provides a abridged tree diagram for the YANG module defined in this document. Annotations used in the diagram are defined in YANG Tree Diagrams [RFC8340].

```
module: ietf-tcp
  +--rw tcp!
     +--rw connections
     |     ...
     +--rw global
     |     ...
     +--ro statistics {statistics}?
           ...
```

4. TCP Configuration YANG Model

Editor's note: How to use ietf-tcp-common as basis? For instance, is the tcp-system-grouping therein needed?

```
<CODE BEGINS> file "ietf-tcp@2020-02-22.yang"

module ietf-tcp {
  yang-version "1.1";
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp";
  prefix "tcp";

  import ietf-yang-types {
    prefix "yang";
    reference
      "RFC 6991: Common YANG Data Types.";
  }
  import ietf-tcp-client {
    prefix "tcpc";
  }
  import ietf-tcp-server {
    prefix "tcps";
  }
  import ietf-tcp-common {
    prefix "tcpcmn";
  }
  import ietf-inet-types {
    prefix "inet";
  }

  organization
    "IETF TCPM Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/tcpm>
     WG List:  <tcpm@ietf.org>

     Authors: Michael Scharf (michael.scharf at hs-esslingen dot de)
              Vishal Murgai (vmurgai at cisco dot com)
```

```
                Mahesh Jethanandani (mjethanandani at gmail dot com)";

    description
      "This module focuses on fundamental and standard TCP functions
       that widely implemented. The model can be augmented to address
       more advanced or implementation specific TCP features.";

    revision "2020-02-22" {
      description
        "Initial Version";
      reference
        "RFC XXX, TCP Configuration.";
    }

    // Features
    feature server {
      description
        "TCP Server configuration supported.";
    }

    feature client {
      description
        "TCP Client configuration supported.";
    }

    feature statistics {
      description
        "This implementation supports statistics reporting.";
    }

    // TCP-AO Groupings

    grouping mkt {
      leaf options {
        type binary;
        description
          "This flag indicates whether TCP options other than TCP-AO
           are included in the MAC calculation. When options are
           included, the content of all options, in the order present,
           is included in the MAC, with TCP-AO's MAC field zeroed out.
           When the options are not included, all options other than
           TCP-AO are excluded from all MAC calculations (skipped over,
           not zeroed).

           Note that TCP-AO, with its MAC field zeroed out, is always
           included in the MAC calculation, regardless of the setting
           of this flag; this protects the indication of the MAC length
           as well as the key ID fields (KeyID, RNextKeyID). The option
```

```
           flag applies to TCP options in both directions
           (incoming and outgoing segments).";
        reference
          "RFC 5925: The TCP Authentication Option.";
      }

      leaf key-id {
        type uint8;
        description
          "An unsigned 1-byte field indicating the Master Key Tuple
           (MKT) used to generate the traffic keys that were used to
           generate the MAC that authenticates this segment.";
        reference
          "RFC 5925: The TCP Authentication Option.";
      }

      leaf rnext-key-id {
        type uint8;
        description
          "An unsigned 1-byte field indicating the MKT that is
           ready at the sender to be used to authenticate received
           segments, i.e., the desired 'receive next' key ID.";
      }
      description
        "A Master Key Tuple (MKT) describes TCP-AO properties to be
         associated with one or more connections.";
    }

    grouping ao {
      leaf enable-ao {
        type boolean;
        default "false";
        description
          "Enable support of TCP-Authentication Option (TCP-AO).";
      }

      leaf send-id {
        type uint8 {
          range "0..255";
        }
        must "../enable-ao = 'true'";
        description
          "The SendID is inserted as the KeyID of the TCP-AO option
           of outgoing segments.";
        reference
          "RFC 5925: The TCP Authentication Option.";
      }
```

```
      leaf recv-id {
        type uint8 {
          range "0..255";
        }
        must "../enable-ao = 'true'";
        description
          "The RecvID is matched against the TCP-AO KeyID of incoming
           segments.";
        reference
          "RFC 5925: The TCP Authentication Option.";
      }

      leaf include-tcp-options {
        type boolean;
        must "../enable-ao = 'true'";
        description
          "Include TCP options in HMAC calculation.";
      }

      leaf accept-ao-mismatch {
        type boolean;
        must "../enable-ao = 'true'";
        description
          "Accept packets with HMAC mismatch.";
      }
      description
        "Authentication Option (AO) for TCP.";
      reference
        "RFC 5925: The TCP Authentication Option.";
    }

    grouping md5 {
      description
        "Grouping for use in authenticating TCP sessions using MD5.";
      reference
        "RFC 2385: Protection of BGP Sessions via the TCP MD5
         Signature.";

      leaf enable-md5 {
        type boolean;
        default "false";
        description
          "Enable support of MD5 to authenticate a TCP session.";
      }
    }

    // Congestion control
```

```
    identity congestion-control-algorithm {
      description
        "Base identity from which all congestion control
         algorithms are derived.";
    }

    identity reno {
      base congestion-control-algorithm;
      description
        "Standard TCP congestion control referred to as
         'Reno algorithm'";
      reference
        "RFC 5681: TCP Congestion Control";
    }

    identity new-reno {
      base congestion-control-algorithm;
      description
        "NewReno modification to TCP's fast recovery algorithm";
      reference
        "RFC 6582: The NewReno Modification to TCP's Fast Recovery
         Algorithm";
    }

    identity ledbat {
      base congestion-control-algorithm;
      description
        "Low Extra Delay Background Transport (LEDBAT)
         congestion control";
     reference
        "RFC 6817: Low Extra Delay Background Transport (LEDBAT)";
    }

    identity dctcp {
      base congestion-control-algorithm;
      description
        "TCP Congestion Control for Data Centers (DCTCP)
         congestion control";
      reference
        "RFC 8257: Data Center TCP (DCTCP): TCP Congestion
         Control for Data Centers";
    }

    identity cubic {
      base congestion-control-algorithm;
      description
        "CUBIC congestion control";
      reference
```

```
        "RFC 8312: CUBIC for Fast Long-Distance Networks";
    }

    grouping tcp-global {
      description
        "Important global TCP stack parameters.";
      leaf mss-max {
        type uint16;
        description
          "Sets the max segment size for TCP connections.";
        reference
          "RFC 1122: Requirements for Internet Hosts -- Communication
           Layers";
      }

      leaf mtu-discovery-enable {
        type boolean;
        description
          "Turns path mtu discovery for TCP connections on (true) or
           off (false)";
        reference
          "RFC 4821: Packetization Layer Path MTU Discovery";
      }

      leaf sack-enable {
        type boolean;
        description
          "Enable negotiation of Selective Acknowledgements (SACK)";
        reference
          "RFC 2018: TCP Selective Acknowledgment Options";
      }

      leaf timestamps-enable {
        type boolean;
        description
          "Enable negotiation of timestamps";
        reference
          "RFC 7323: TCP Extensions for High Performance";
      }

      leaf window-scale-enable {
        type boolean;
        description
          "Enable negotiation of receive window scaling";
        reference
          "RFC 7323: TCP Extensions for High Performance";
      }
```

```
      leaf ecn-enable {
        type enumeration {
          enum disable;
          enum passive;
          enum active;
        }
        description
          "Enabling of Explicit Congestion Notification (ECN).";
        reference
          "RFC 3168: The Addition of Explicit Congestion
           Notification (ECN) to IP";
      }

      leaf fin-timeout {
        type uint16;
        units "seconds";
        description
          "When a connection is closed actively, it must linger in
           TIME-WAIT state for a time 2xMSL (Maximum Segment Lifetime).
           This parameter sets the TIME-WAIT timeout duration in
           seconds.";
        reference
           "RFC 793: Transmission Control Protocol";
      }

      leaf congestion-control-default {
        type identityref {
          base congestion-control-algorithm;
        }
        description
          "This parameter selects the congestion control algorithm that
           is used by default for new TCP connections. The default may
           be overridden per connection by means outside the scope of
           this model (e.g., via the Sockets API).";
         reference
           "RFC 2914: Congestion Control Principles";
      }
    }

    // TCP configuration

    container tcp {
      presence "The container for TCP configuration.";

      description
        "TCP container.";

      container connections {
```

```
        list connection {
          key "local-address remote-address local-port remote-port";

          leaf local-address {
            type inet:ip-address;
            description
              "Local address that forms the connection identifier.";
          }

          leaf remote-address {
            type inet:ip-address;
            description
              "Remote address that forms the connection identifier.";
          }

          leaf local-port {
            type inet:port-number;
            description
              "Local TCP port that forms the connection identifier.";
          }

          leaf remote-port {
            type inet:port-number;
            description
              "Remote TCP port that forms the connection identifier.";
          }

          container common {
            uses tcpcmn:tcp-common-grouping;

            leaf congestion-control {
              type identityref {
                base congestion-control-algorithm;
              }
              config false;
              description
                "Type of the actually used TCP congestion control
                 algorithm. It may be different from the default
                 algorithm, for instance, if an application has
                 explicitly selected an algorithm.";
              reference "RFC 2914: Congestion Control Principles";
            }

            choice authentication {
              case ao {
                uses ao;
                description
                  "Use TCP-AO to secure the connection.";
```

```
          }

          case md5 {
            uses md5;
            description
              "Use TCP-MD5 to secure the connection.";
          }
          description
            "Choice of how to secure the TCP connection.";
        }
        leaf tcp-nodelay {
          type boolean;
          config false;
          description
            "Disabling of the 'Nagle algorithm'.";
        }
        description
          "Common definitions of TCP configuration. This includes
           parameters such as how to secure the connection,
           keepalives and idle time, that can be part of either
           the client or server.";
      }

      container server {
        if-feature server;
        uses tcps:tcp-server-grouping;
        description
          "Definitions of TCP server configuration.";
      }

      container client {
        if-feature client;
        uses tcpc:tcp-client-grouping;
        description
          "Definitions of TCP client configuration.";
      }
      description
        "Connection related parameters.";
    }
    description
      "A container of all TCP connections.";
  }

  container global {
    uses tcp-global;
    description
      "Parameters affecting all TCP connections";
  }
```

```
      container statistics {
        if-feature statistics;
        config false;

        leaf active-opens {
          type yang:counter32;
          description
            "The number of times that TCP connections have made a direct
             transition to the SYN-SENT state from the CLOSED state.";
        }

        leaf passive-opens {
          type yang:counter32;
          description
            "The number of times TCP connections have made a direct
             transition to the SYN-RCVD state from the LISTEN state.";
        }

        leaf attempt-fails {
          type yang:counter32;
          description
            "The number of times that TCP connections have made a direct
             transition to the CLOSED state from either the SYN-SENT
             state or the SYN-RCVD state, plus the number of times that
             TCP connections have made a direct transition to the
             LISTEN state from the SYN-RCVD state.";
        }

        leaf establish-resets {
          type yang:counter32;
          description
            "The number of times that TCP connections have made a direct
             transition to the CLOSED state from either the ESTABLISHED
             state or the CLOSE-WAIT state.";
        }

        leaf currently-established {
          type yang:gauge32;
          description
            "The number of TCP connections for which the current state
             is either ESTABLISHED or CLOSE-WAIT.";
        }

        leaf in-segments {
          type yang:counter32;
          description
            "The total number of segments received, including those
             received in error.  This count includes segments received
```

```
             on currently established connections.";
        }

        leaf out-segments {
          type yang:counter32;
          description
            "The total number of segments sent, including those on
             current connections but excluding those containing only
             retransmitted octets.";
        }

        leaf retransmitted-segments {
          type yang:counter32;
          description
            "The total number of segments retransmitted; that is, the
             number of TCP segments transmitted containing one or more
             previously transmitted octets.";
        }

        leaf in-errors {
          type yang:counter32;
          description
            "The total number of segments received in error (e.g., bad
             TCP checksums).";
        }

        leaf out-resets {
          type yang:counter32;
          description
            "The number of TCP segments sent containing the RST flag.";
        }

        action reset {
          description
            "Reset statistics action command.";
          input {
            leaf reset-at {
              type yang:date-and-time;
              description
                "Time when the reset action needs to be
                 executed.";
            }
          }
          output {
            leaf reset-finished-at {
              type yang:date-and-time;
              description
                "Time when the reset action command completed.";
```

```
            }
          }
        }
        description
          "Statistics across all connections.";
      }
    }
  }

  <CODE ENDS>
```

5. IANA Considerations

5.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in IETF XML Registry [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-tcp
Registrant Contact: The TCPM WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

5.2. The YANG Module Names Registry

This document registers a YANG modules in the YANG Module Names registry YANG - A Data Modeling Language [RFC6020]. Following the format in YANG - A Data Modeling Language [RFC6020], the following registrations are requested:

```
name:         ietf-tcp
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp
prefix:       tcp
reference:    RFC XXXX
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or

RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

7. References

7.1. Normative References

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", draft-ietf-netconf-tcp-client-server-03 (work in progress), October 2019.

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

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[I-D.ietf-dots-data-channel]
Boucadair, M. and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-31 (work in progress), July 2019.

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-07 (work in progress), October 2019.

[I-D.ietf-taps-interface]
Trammell, B., Welzl, M., Enghardt, T., Fairhurst, G., Kuehlewind, M., Perkins, C., Tiesel, P., Wood, C., and T. Pauly, "An Abstract Application Layer Interface to Transport Services", draft-ietf-taps-interface-05 (work in progress), November 2019.

[RFC4022] Raghunarayan, R., Ed., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, DOI 10.17487/RFC4022, March 2005, <https://www.rfc-editor.org/info/rfc4022>.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, DOI 10.17487/RFC4898, May 2007, <https://www.rfc-editor.org/info/rfc4898>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC6643] Schoenwaelder, J., "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules", RFC 6643, DOI 10.17487/RFC6643, July 2012, <https://www.rfc-editor.org/info/rfc6643>.

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

[RFC8513] Boucadair, M., Jacquenet, C., and S. Sivakumar, "A YANG Data Model for Dual-Stack Lite (DS-Lite)", RFC 8513, DOI 10.17487/RFC8513, January 2019, <https://www.rfc-editor.org/info/rfc8513>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

Appendix A. Acknowledgements

Michael Scharf was supported by the StandICT.eu project, which is funded by the European Commission under the Horizon 2020 Programme.

The following persons have contributed to this document by reviews: Mohamed Boucadair

Appendix B. Changes compared to previous versions

Changes compared to draft-scharf-tcpm-yang-tcp-02

o Initial proposal of a YANG model including base configuration parameters, TCP-AO configuration, and a connection list

o Editorial bugfixes and outdated references reported by Mohamed Boucadair

o Additional co-author Mahesh Jethanandani

Changes compared to draft-scharf-tcpm-yang-tcp-01

- Alignment with [I-D.ietf-netconf-tcp-client-server]

- Removing backward-compatibility to the TCP MIB

- Additional co-author Vishal Murgai

Changes compared to draft-scharf-tcpm-yang-tcp-00

- Editorial improvements

Authors' Addresses

Michael Scharf
Hochschule Esslingen - University of Applied Sciences
Flandernstr. 101
Esslingen 73732
Germany

Email: michael.scharf@hs-esslingen.de

Vishal Murgai
Cisco Systems Inc

Email: vmurgai@cisco.com

Mahesh Jethanandani
VMware

Email: mjethanandani@gmail.com