Status update on draft-ietf-tcpm-yang-tcp

Michael Scharf

Vishal Murgai

Mahesh Jethanandani

With contributions from Simon Bauer and Martin Mager

TCPM @ IETF 110

Scope and status

- **Basic YANG model** for TCP configuration
- **Narrow scope**
  1. TCP basic statistics (optional)
  2. TCP connection list

  Established by TCP MIB

  3. TCP-AO and TCP MD5 with TCP-AO being strongly RECOMMENDED

  For instance, for draft-ietf-idr-bgp-model

  4. Import of groupings from draft-ietf-netconf-tcp-client-server
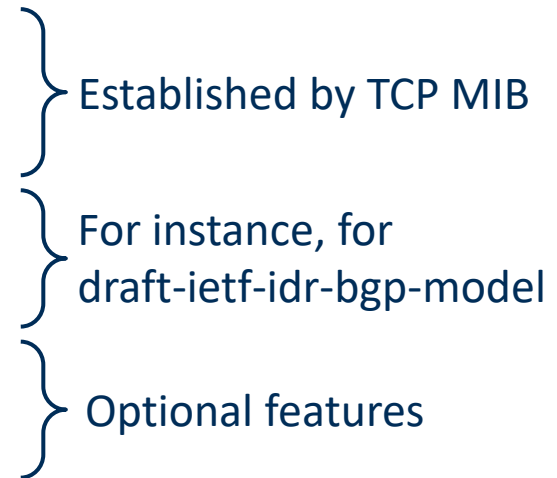
  Optional features
- **No major I-D changes recently**
  - Previously known issues were addressed
  - Milestone date to be aligned with draft-ietf-idr-bgp-model (e.g., end of 2021)
  - Running code needed

Relationship between YANG models

App
e.g. NETCONF
client/server

draft-ietf-netconf-
tcp-client-server

e.g. socket

TCP/IP stack

draft-ietf-tcpm-
yang-tcp
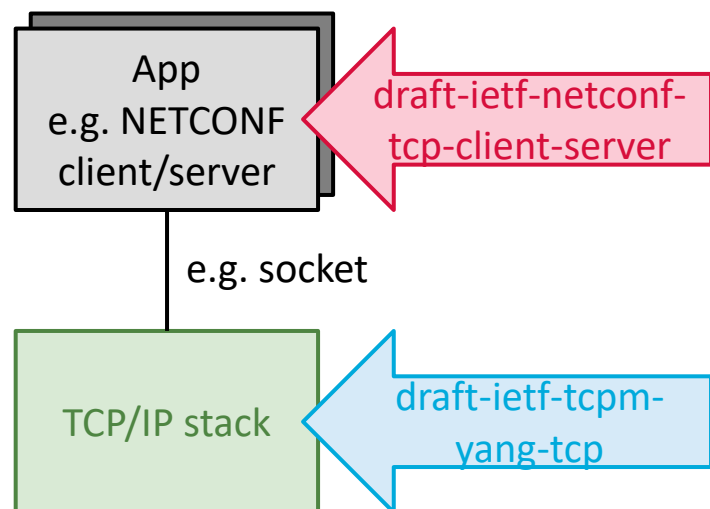
- **draft-ietf-netconf-tcp-client-server**
  - WG document in NETCONF WG as part of a larger set of YANG models
  - Modeling **one connection** as seen by app
  - Example: IP addresses, TCP port numbers
  - Information related to socket interface
- **draft-ietf-tcpm-yang-tcp**
  - WG document in TCPM WG
  - Modeling TCP stack configuration
  - Global view on **all TCP connections**
  - Information related to operating system configuration (e.g., sysctl)
- **Different scopes of YANG models**
  - "User space" vs. "kernel space"
  - Some inherent overlap, e.g., for TCP keep-alive configuration

Compete tree diagram

```
module: ietf-tcp
  +--rw tcp!
     +--rw connections                                    ← Connection list
     |  +--rw connection*
     |          [local-address remote-address local-port remote-port]
     |     +--rw local-address      inet:ip-address
     |     +--rw remote-address     inet:ip-address
     |     +--rw local-port         inet:port-number
     |     +--rw remote-port        inet:port-number
     |     +--rw common
     |        +--rw keepalives!
     |        |  +--rw idle-time         uint16
     |        |  +--rw max-probes        uint16
     |        |  +--rw probe-interval    uint16
     |        +--rw (authentication)?                     ← TCP-AO (and TCP MD5)
     |           +--:(ao)
     |           |  +--rw enable-ao?            Boolean
     |           |  +--rw send-id?              Uint8
     |           |  +--rw recv-id?              Uint8
     |           |  +--rw include-tcp-options?  Boolean
     |           |  +--rw accept-key-mismatch?  Boolean
     |           +--:(md5)
     |              +--rw enable-md5?           Boolean
     +--rw server {server}?
     |  +--rw local-address     inet:ip-address
     |  +--rw local-port?       inet:port-number
     |  +--rw keepalives!
     |     +--rw idle-time         uint16          ← (optional) server
     |     +--rw max-probes        uint16            and client config.
     |     +--rw probe-interval    uint16
     +--rw client {client}?
     |  +--rw remote-address    inet:host
     |  +--rw remote-port?      inet:port-number
     |  +--rw local-address?    inet:ip-address
     |  +--rw local-port?       inet:port-number
     |  +--rw keepalives!
     |     +--rw idle-time         uint16
     |     +--rw max-probes        uint16
     |     +--rw probe-interval    uint16                          1
```

```
  +--ro statistics {statistics}?                            ← Stats
     +--ro active-opens?             yang:counter32
     +--ro passive-opens?            yang:counter32
     +--ro attempt-fails?            yang:counter32
     +--ro establish-resets?         yang:counter32
     +--ro currently-established?    yang:gauge32
     +--ro in-segments?              yang:counter64
     +--ro out-segments?             yang:counter64
     +--ro retransmitted-segments?   yang:counter32
     +--ro in-errors?                yang:counter32
     +--ro out-resets?               yang:counter32
     +---x reset
        +---w input
        |  +---w reset-at?    yang:date-and-time
        +--ro output
           +--ro reset-finished-at?   yang:date-and-time

                                                            2
```

Ongoing prototyping

- **Prototype** for draft-ietf-tcpm-yang-tcp
  - Ongoing student research project at Hochschule Esslingen – University of Applied Sciences
  - Developers: Simon Bauer and Martin Mager
  - Goal is to evaluate the YANG model
- **Implementation details (work-in-progress)**
  - Basic implementation for Linux
    - As far as possible with the vanilla Linux TCP stack
    - One challenge is the lack of an open source TCP-AO implementation (despite several known closed-source TCP-AO implementations)
  - NETCONF server based on the open-source software "clixon"
    - Clixon source code at https://github.com/clicon/clixon
    - Clixon supports NETCONF, RESTCONF, and CLI interfaces
    - No NMDA support
- **Planned open-source release** of the final prototype

Finding: Write access to connection list

```
module: ietf-tcp
  +--rw tcp!
     +--rw connections
     |  +--rw connection*
     |          [local-address remote-address local-port remote-port]
     |  ...
```

- **Issue**: Connection list modeled as read-write
  - Read-write access is needed because of YANG semantics (keys are not possible on read-only list)
  - Otherwise, nodes in the list are read-only
  - Not meant to allow creation of new TCP connection via the YANG model (i.e., from outside any application)
  - Similar issue in interfaces YANG model (RFC 8343)
- **Proposed solution**: Better explanation in description inside the model

Finding: Client/server imports

```
|  ...
+--rw server {server}?
|  ...
+--rw client {client}?
|  ...
```

- **Issue**: Client/server from draft-ietf-netconf-tcp-client-server
  - Enables endpoint configuration (i.e., IP address and port number)
  - Optional to implement and enabled by two YANG features
  - No clear use case for this inside configuration of TCP stack
- **Proposed solution**: Remove imports and the two features for client and server configuration
  - Simplifies the model
  - Reduced dependencies between NETCONF and TCPM I-Ds
  - Client or server configuration more straightforward in YANG models for applications (e.g., using draft-ietf-netconf-tcp-client-server)

Finding: Reset RPC for statistics

```
+--ro statistics {statistics}?
|  ...
+---x reset
    +---w input
    |  +---w reset-at? yang:date-and-time
    +--ro output
        +--ro reset-finished-at? yang:date-and-time
```

- **Issue**: Resetting TCP stats is not always supported
  - Resetting stats is typically possible in router operating systems
  - Support in some host OS (e.g., "netstat –s –z" in FreeBSD)
  - However, not easily possible in vanilla Linux kernels
- **To be discussed**: Make reset RPC optional?
  - RPC support could be an optional feature in YANG model
  - An alternative would be just to keep the reset RPC as-is

Feedback from Juniper and Nokia

```
|            +--:(ao)
|            |  +--rw enable-ao?          Boolean
|            |  +--rw send-id?            Uint8
|            |  +--rw recv-id?            Uint8
|            |  +--rw include-tcp-options?  Boolean
|            |  +--rw accept-key-mismatch?  Boolean
```

- Offlist feedback based on lessons learnt during a TCP-AO interop between Juniper and Nokia routers
  - Melchior Aelmans <maelmans@juniper.net>
  - Greg Hankins <greg.hankins@nokia.com>
- **Issue**: Clear description of how to set send-id and recv-id
  - Send-id and recv-id must be configured from the perspective of each endpoint
  - Values of send-id/receive-id need to match each other
  - Example: send-id on endpoint A needs to match receive-id on endpoint B, and vice versa
- **Proposed solution**: Better explanation of how to use send-id and recv-id in the YANG model