Management information base for MPTCP

Fabien Duchêne <fabien.duchene@uclouvain.be>
Christoph Paasch <cpaasch@apple.com>
Olivier Bonaventure <olivier.bonaventure@uclouvain.be>

Why ?

A better understanding of MTCP operations:

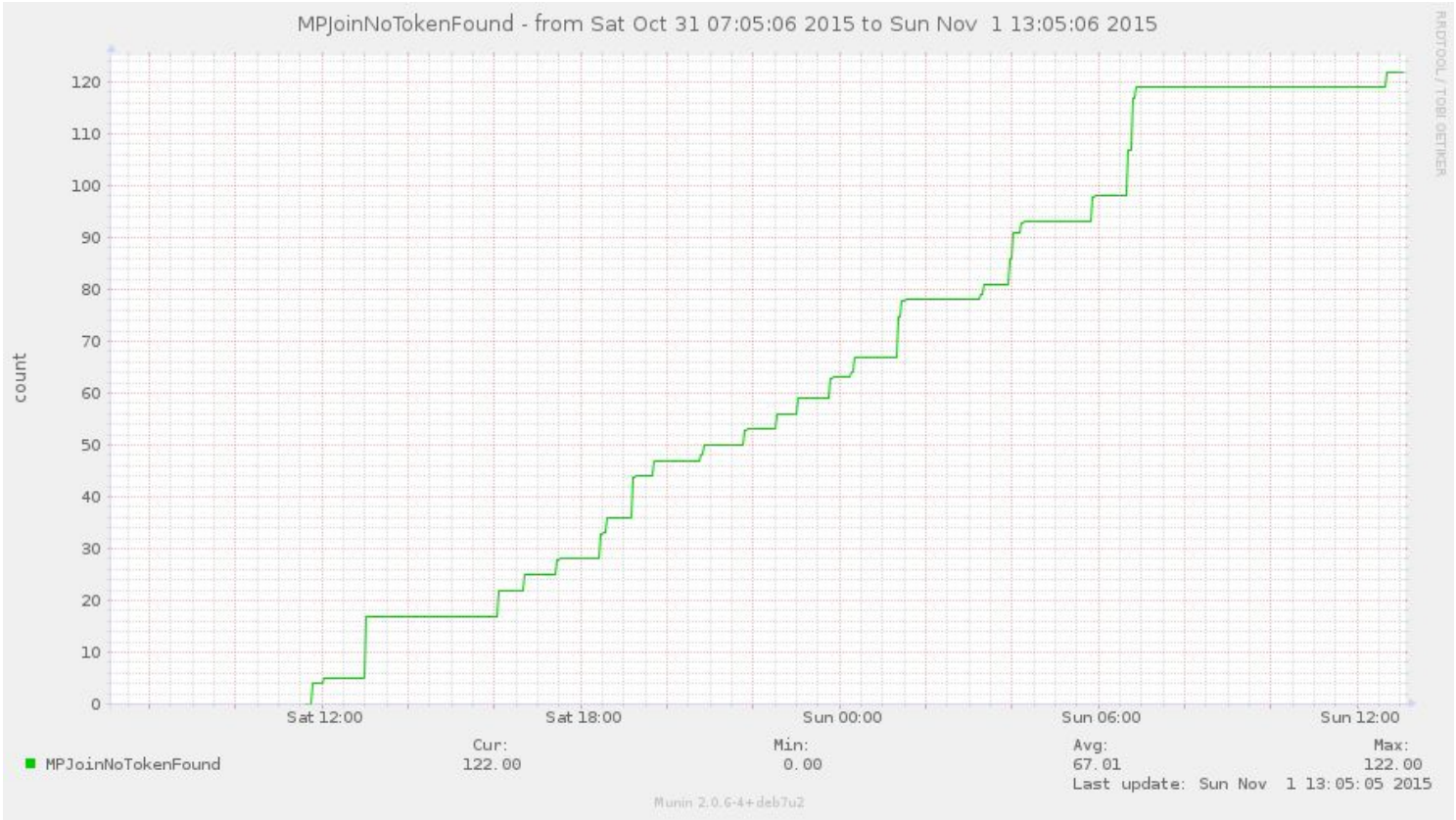
- **Troubleshooting**: the MIB covers the different failure conditions
- **Statistics**: track the transmission and reception of data at the MPTCP-layer

Current implementation

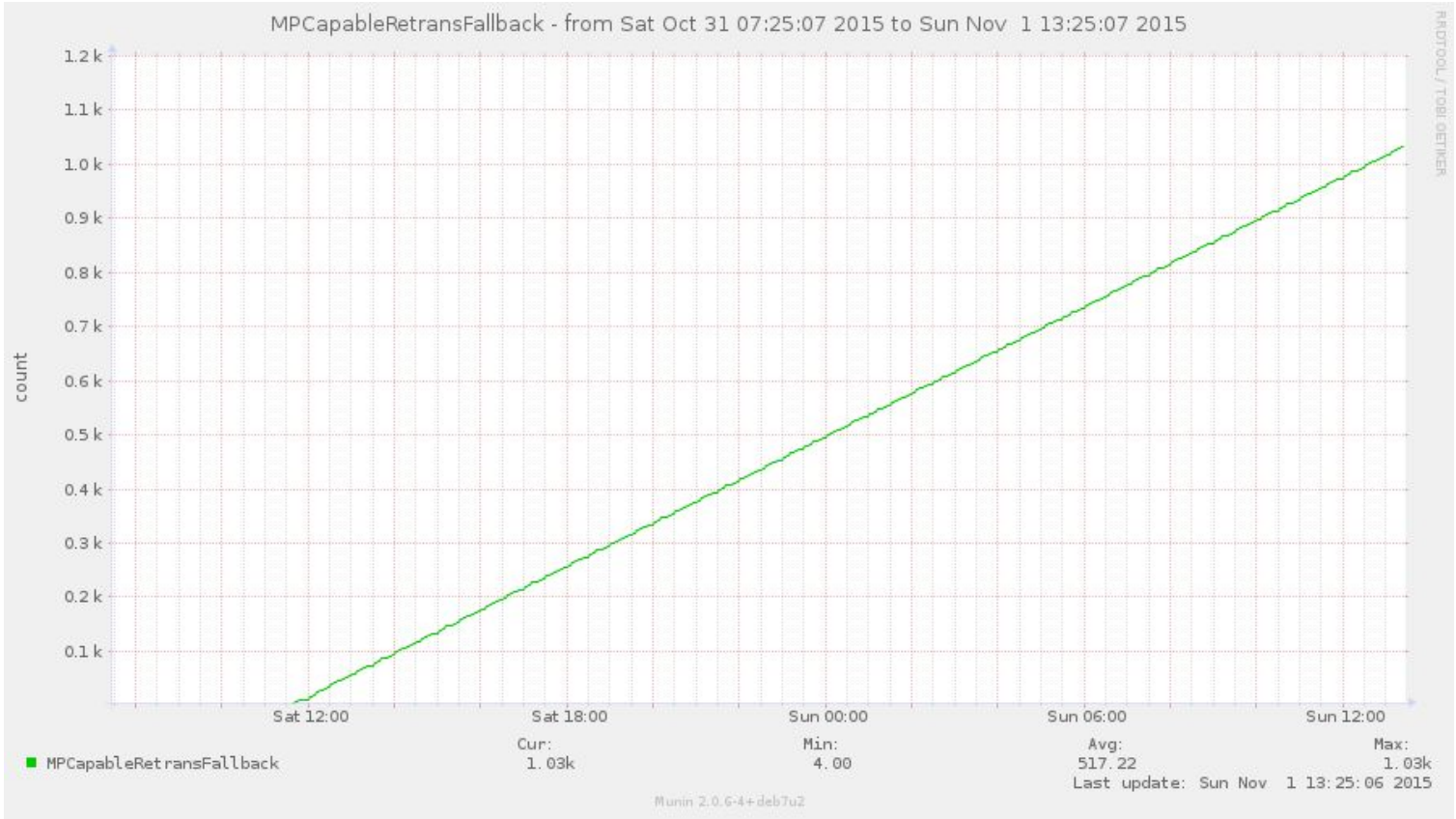
The Linux implementation uses 40 counters:

```
[….]
MPTCP_MIB_JOINSYNTX,          /* Sent a SYN + MP_JOIN */
MPTCP_MIB_JOINSYNRX,          /* Received a SYN + MP_JOIN */
MPTCP_MIB_JOINSYNACKRX,           /* Received a SYN/ACK + MP_JOIN */
MPTCP_MIB_JOINSYNACKMAC, /* HMAC was wrong on SYN/ACK + MP_JOIN */
MPTCP_MIB_JOINACKRX,          /* Received an ACK + MP_JOIN */
[….]
```

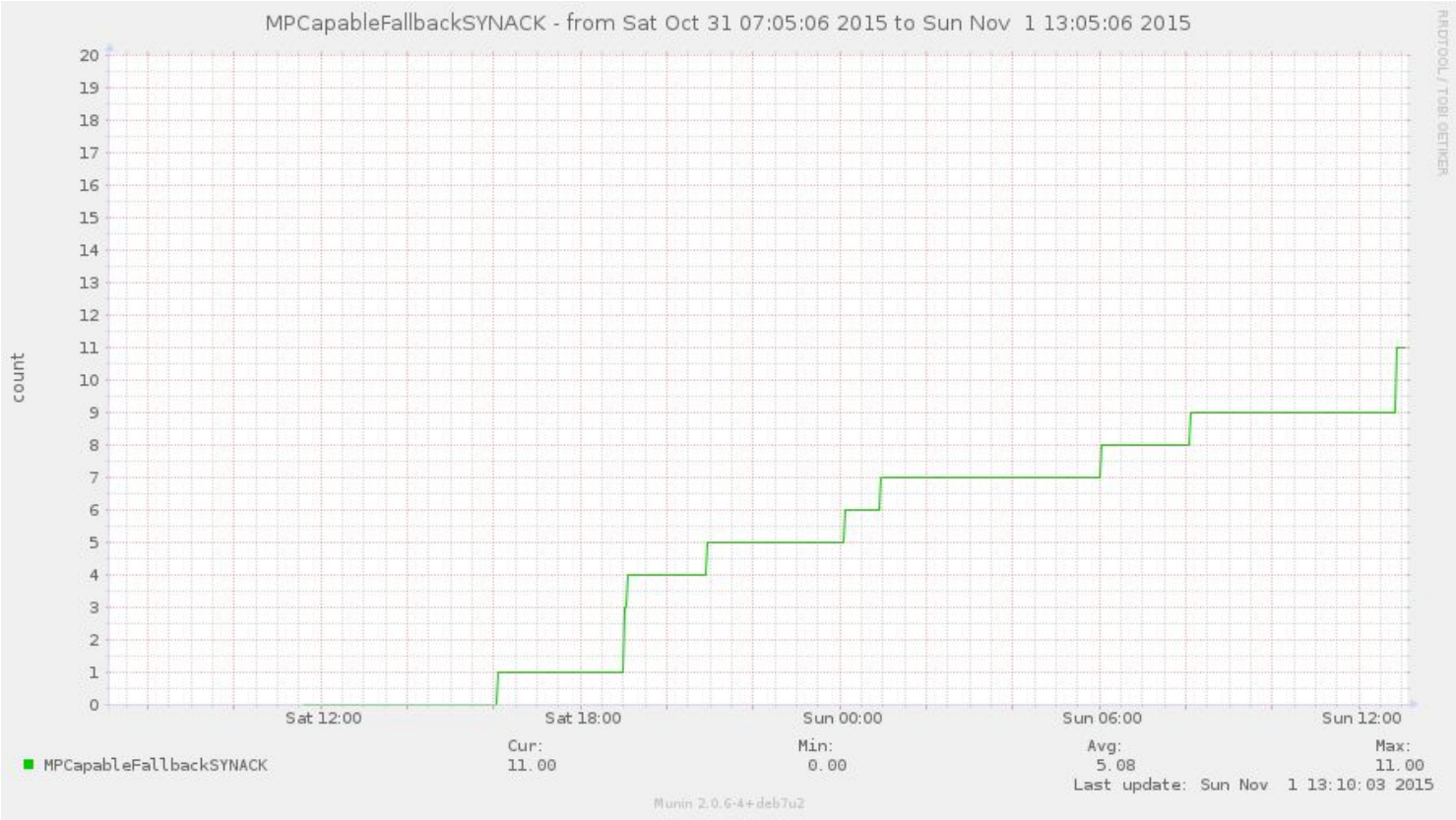
Troubleshooting



Troubleshooting



Troubleshooting



Too descriptive ?

The current implementation is **very** descriptive:

- ⇨ Confusing if you are not familiar with the implementation
- ⇨ Might not apply to other implementations

The right balance

Find the right “balance” without being :

- **Too descriptive**: too many counters
- **Too vague**: doesn’t help understanding

The proposed draft contains 20 counters.

Example: the failures

In the draft we splitted the failures scenarios :

- **Fallback while “probing”**: SYN+ACK without CAPABLE/JOIN, ACK without DATA_ACK,...
- **Fallback when established**: DSS-checksum, too many sgements without DSS mapping,...
- **Attacks?**: no token, bad HMAC, ….
- **Other failures**: bad DSS mapping,...

Example: the failures

mptcpFailedToEstablishInitialSubflows OBJECT-TYPE

```
SYNTAX     Counter
UNITS      "connections"
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
       "The number of initial MPTCP subflows (i.e. the initial SYN
    segment contained the MP_CAPABLE option) that could not
        transition to the ESTABLISHED state from the SYN-RECEIVED
        or SYN-SENT states. The reason being one of:
        - the SYN+ACK didn't contain a MP_CAPABLE
        - the first ACK didn't contain a DATA_ACK or the first
    data-segment did not contain a DSS mapping
        - 4-way handshake didn't complete (SYN+ACK or ACK not received)
    Given these reasons, a connection could not get established or fell
    back to regular TCP. They are most likely due to middleboxes
    interfering with the connection."
::= { mptcp 9 }
```

Example: the failures

mptcpFallbackEstablishedConnections OBJECT-TYPE

```
SYNTAX     Counter
UNITS      "connections"
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
       "The number of MPTCP connections that fell back to regular TCP
        while being already ESTABLISHED. The reason being one of:
        - Reception of more than a window worth of data without DSS
        - Reception of a segment with an incorrect DSS checksum
    This happens when a middlebox is interfering with the data
    flow after the connection has been successfully established."
::= { mptcp 11 }
```

Example: traffic statistics

Currently, 3 counters :

- **mptcpReceivedInOrder**: The number of segments that were received in order at the MPTCP (meta) level.
- **mptcpReceivedOutOfOrder**: The number of segments that were received out of order at the MPTCP (meta) level.
- **mptcpSentSegments**: The number of segments that were emitted at the MPTCP (meta) level.

Conclusion

- A MIB is useful to give a better understanding
- We need to find the right balance between being:
  - Too descriptive
  - Too vague
- Ideas ?