intdash®

White Paper

iSCP (intdash Stream Control Protocol)

30th May, 2018

aptpod

iSCP (intdash Stream Control Protocol)

iSCP (intdash Stream Control Protocol) is a protocol that aptpod originally developed and that is used for internal communication of modules to realize each function of intdash. iSCP is used for data streaming between a relay broker of streaming data and the module of an edge device to deliver the main functions of intdash, such as complete collection, efficient transmission, and flow control.

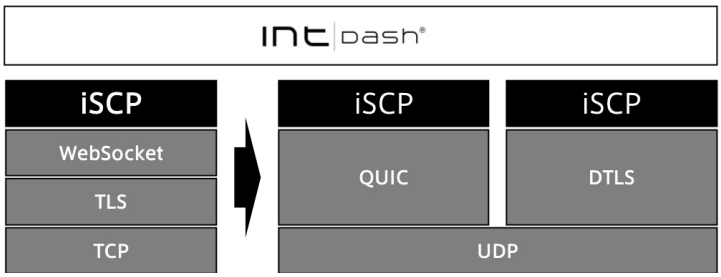
Protocol Stack

The current version of iSCP uses WebSocket Secure as transport. The advantages of employing WebSocket are as follows:

- Bidirectional socket communication almost same as TCP/TLS is possible
- Path compression function supported at protocol level
- HTTP port used, making the protocol hard to be cut off by proxy or firewall
- Development cost can be lowered by using services for web systems

The disadvantage is that reduction of transmission efficiency by a header is suppressed by the function of transmission buffering equipped to iSCP, even though there are points where a WebSocket header is appended as extra overhead.

However, WebSocket is a TCP-based protocol and not an optimum solution for use cases with strict requirements on delay, such as teleoperation. aptpod expects that such strict requirements will increase and has started an investigation and study on using QUIC, a new protocol based on UDP, and UDP/DTLS as transport.



Data Format

As data structures for transmission, a **Unit** structure that indicates the data itself that has been generated, a **Section** structure for complete collection, and a **Flush** structure for efficient transmission are important.

Unit Structure

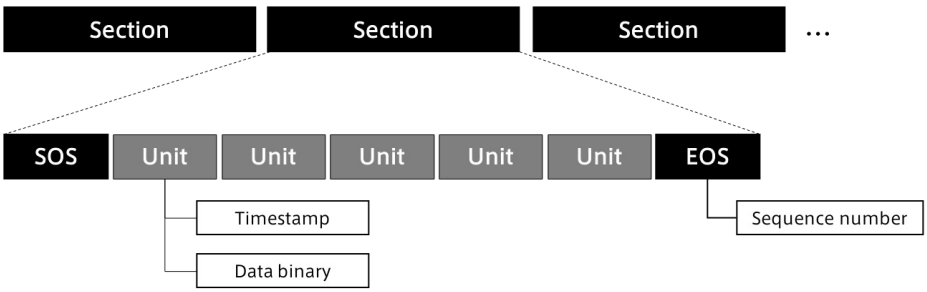
All time-series data that have been generated are stored in a structure called a Unit. A Unit structure consists of a stream ID to identify a stream, elapsed time indicating time of occurrence, payload, which is the main body of the data, and other detailed meta information.

Section Structure

A unit storing time series data becomes a section when a marker called Section Marker (SOS and EOS[1]) is attached to the front and back. This section is successively streamed through a secure communication path formed by WebSocket Secure.

A section is a structure for a complete collection, as mentioned above, and a sequential number that is used to check a loss is stored in the Section Markers to be appended. A server that has received a Section returns ACK to the edge device after completion of persistence, completing a sequence for completely collecting data that has been generated.

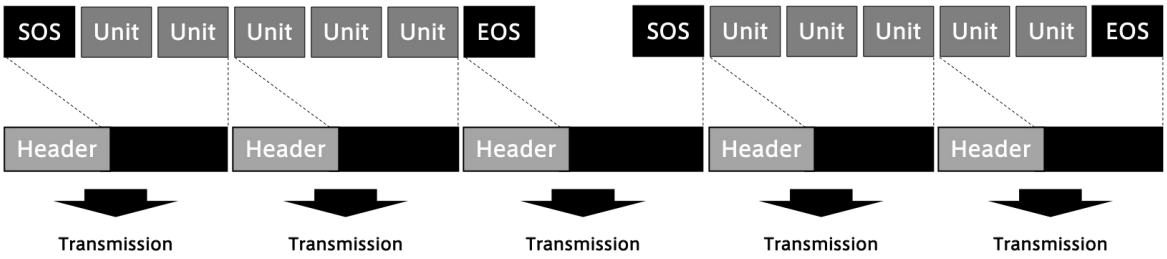
It can be expected, by lengthening a Section, that the processing load of the edge device and server can be alleviated by reducing the number of times of returning and checking ACK and the number of times of I/O processing for persistence. On the other hand, there are disadvantages such that the unit in which data must be retransmitted in case a loss occurs increases and that the retransmission queue of the edge device and the buffer of the server tend to become long. It is necessary to decide in accordance with the requirements of the application how many units are combined into a section.



Flush Structure

When a section is transmitted over WebSocket Secure, an enumeration of units and section markers is actually transmitted. Flush is a unit of transmission to buffer these units and section markers for a specific period and transmit them all at once.

By using a long flush, improvements of the transmission efficiency, such as a decrease in the number of transmission/reception requests to the network and a decline in the overhead owing to a header, can be expected. On the other hand, the length of a flush, like that of a section, must be set according to the requirements of the application because a buffering delay is added.

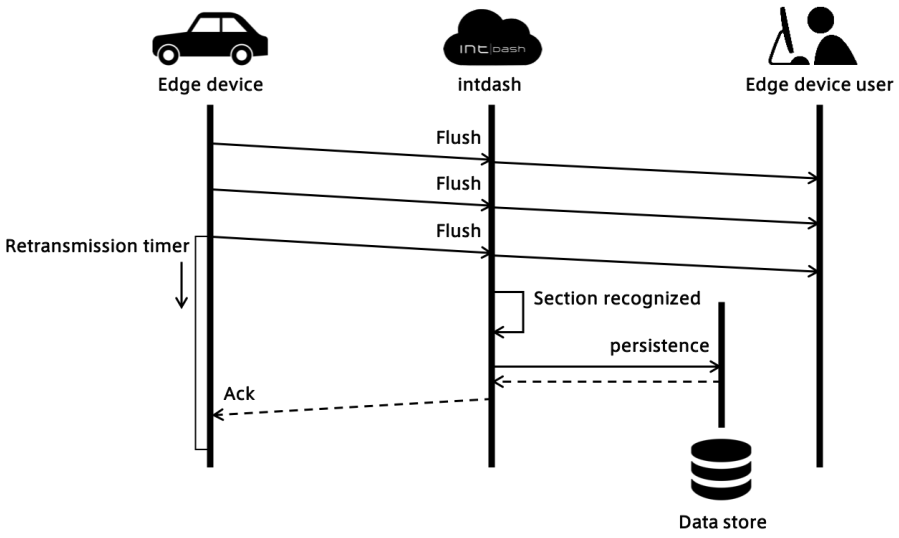


Complete Collection Sequence

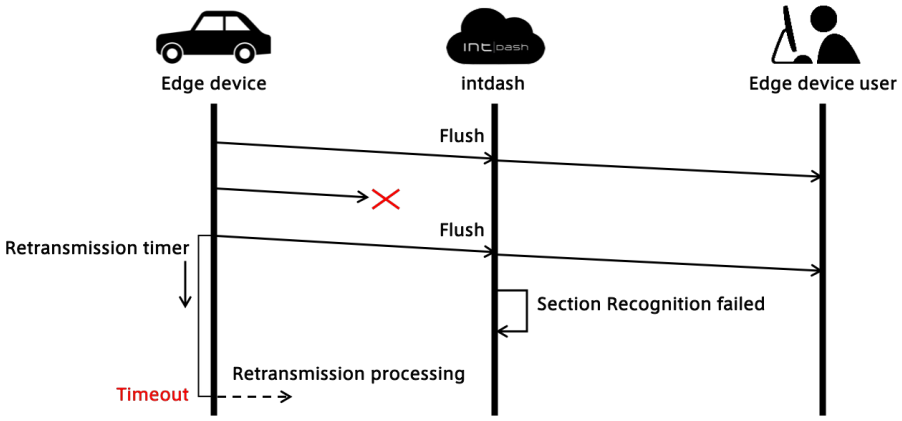
Complete collection with iSCP is materialized by checking arrival at a data store in section units.

At the side of an edge device, data that have been generated is buffered and put together into a flush and sent to the server. A server that has received the flush immediately transfers it to the receiver edge device and, at the same time, sequentially checks units and section markers that were combined into the flush to extract sections. When sections have been extracted, all units contained are persisted to a data store and Section ACK is returned to the edge device.

Meanwhile, the edge device monitors the sections transmitted to the server for returning of Section ACK until the specified time has elapsed. For sections to which Section ACK was not returned even after the time has elapsed, the contained units are moved to a retransmitted stream. The retransmitted stream periodically monitors units that are to be retransmitted and executes retransmission if there are units that must be retransmitted.



Note that, although iSCP guarantees the integrity of data to be persisted, it does not guarantee the integrity of data to be streamed in real-time. The current version of iSCP assumes use of WebSocket Secure for transfer. It is therefore expected to make up for a loss and guarantee a sequence in a manner equivalent to TCP/TLS, but does not perform any additional processing to bring transmission delay under control. It should also be noted that a flow of data may be intentionally suppressed by a dynamic filter described below.



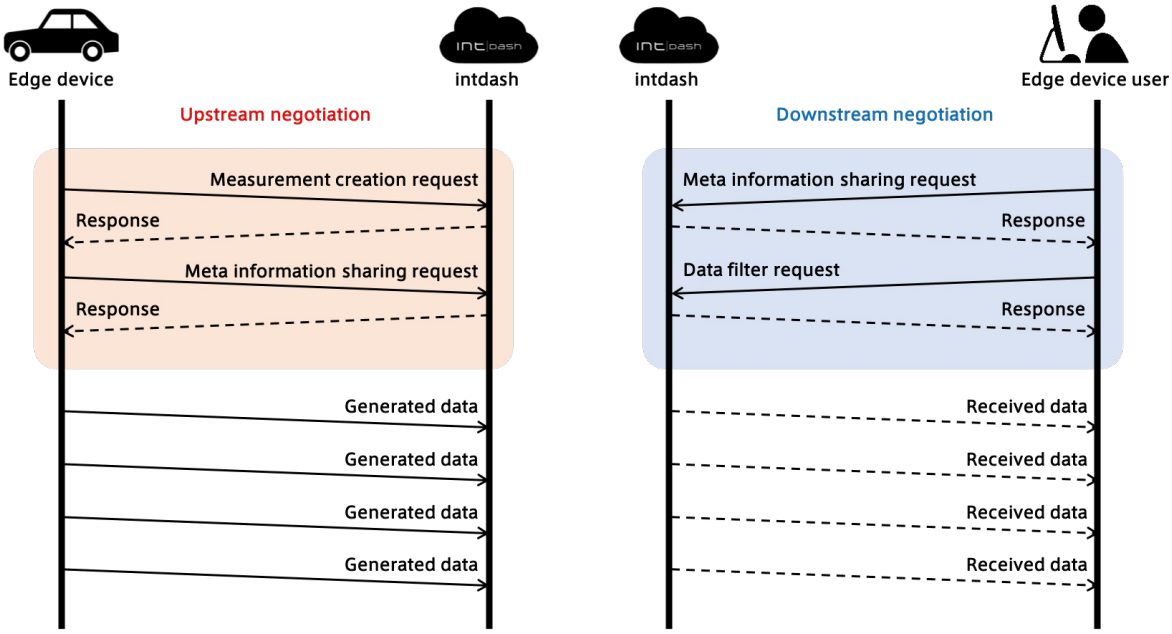
Negotiation Sequence

iSCP defines a procedure of negotiation that is carried out before the start of transmission or reception. By negotiation, meta information that has not changed during communication is shared between the transmitter and receiver and the quantity of data to be transmitted during streaming is reduced.

Negotiation comes in two types: upstream negotiation that is used in the direction of transmission from an edge device to a broker and downstream negotiation that is used in the direction of transmission from the broker to the edge device.

Upstream negotiation generates new measurement as necessary. Creating measurement is not essential for starting streaming as existing measurement data. When creation of measurement is over, a static meta information sharing sequence is started and a stream is opened, sharing the measurement ID, transmission source edge ID, transmission destination edge IDs (plural), persistence need flag, and retransmission flag.

Downstream negotiation first opens a stream by sharing the transmission source edge ID and transmission destination edge IDs in a static meta information sharing sequence. After that, the type of the data transmitted from the server in each stream is set in a received data filtering setting sequence.



The upstream or downstream that has been opened by negotiation is distinguished at the reception side by a stream ID so that it can be transmitted multiplexed into one connection.