# Quick Overview

There will be two ports used on the multicast group and a third TCP port. The first port is for the actual multicast application data, and outside the scope of this document. The second port is for locating the floating master and for voting on the floating master. The TCP port is used for one-to-one authentication with TLS and key provisioning.



# Floating Master

This is being used in an environment where there cannot be a single master device, no single point of failure is required. Authentication and key generation requires a master device.

Both RFC4046[[1]](#footnote-1) and KEOH[[2]](#footnote-2) discuss having a single master device that all group members will contact via unicast to authenticate themselves with a standard DTLS handshake. The multicast keys are provided from the master to the new peer.

Since there cannot be a single master device, one of the endpoints must generate the group keys and distribute them to the peers, and any of the peers may try to be master. If another device also thinks it is master, the higher multicast group ID peer shall stop trying to be the master.

The floating master will listen for a beacon message from any new devices wishing to join. The current master shall send a message of its own to identify itself to the multicast group.

When there is no master and no shared keys, all devices shall wait their member ID in seconds before attempting to assert being the master.

While the peer with the lowest member ID wins when multiple peers try to assert being master, if a member joins and it has an ID lower than the current master, the new member shall join and shall not try to assert being the floating master.

Should multiple peers all try to connect to the floating master at the same time, for example when all are powered up at the same time, the floating master may at its discretion ignore many of the incoming TLS connections. The other devices will retry their connection attempts with an ever increasing timeout.

# Joining the Group

When a new joining peer discovers the master, the peer shall make a TCP connection to the master on the group key port. The master and peer will perform a TLS handshake, using the pre-shared key programmed into both, or with mutual authentication with certificates in the future. The master will send the new peer the current keying state. At this point the new peer can start sending to and receiving from the data group.



# Rekeying

Rekeying can be triggered by any peer in the association with a request message on the keying group. Keying shall be triggered by the floating master when reaching a sequence number sum “high-water mark.”

When the rekey is triggered, the floating master will generate new keying material and transmit a new key notice to all the peers. All peers will need to connect to the master via TLS and get the new keys. All peers shall transmit a change cipher spec message when accepting the new keys. All peers will keep the old key around in case messages from the previous epoch are received. The old epoch should be ignored after several seconds. (Configurable, based on number of messages received in the new Epoch.) The beacon with the new keying material will be retransmit on a fast interval that slows down to make sure all peers get it. (This is wrong. This will be all the peers contacting the floating master After several transmits of the new keys, the master will start using the new keys. (When everybody knows everybody has the new keys, switch over.)

The epoch number shall be the same across all peers, and the floating master will set it with the keys.

If a device misses the new keys announcement, it should start seeing change cipher spec messages from its peers and will know it missed something. It may open a connection to the master and request the current key.

1. <https://tools.ietf.org/html/rfc4046> [↑](#footnote-ref-1)
2. <https://tools.ietf.org/html/draft-keoh-dice-multicast-security-08> [↑](#footnote-ref-2)