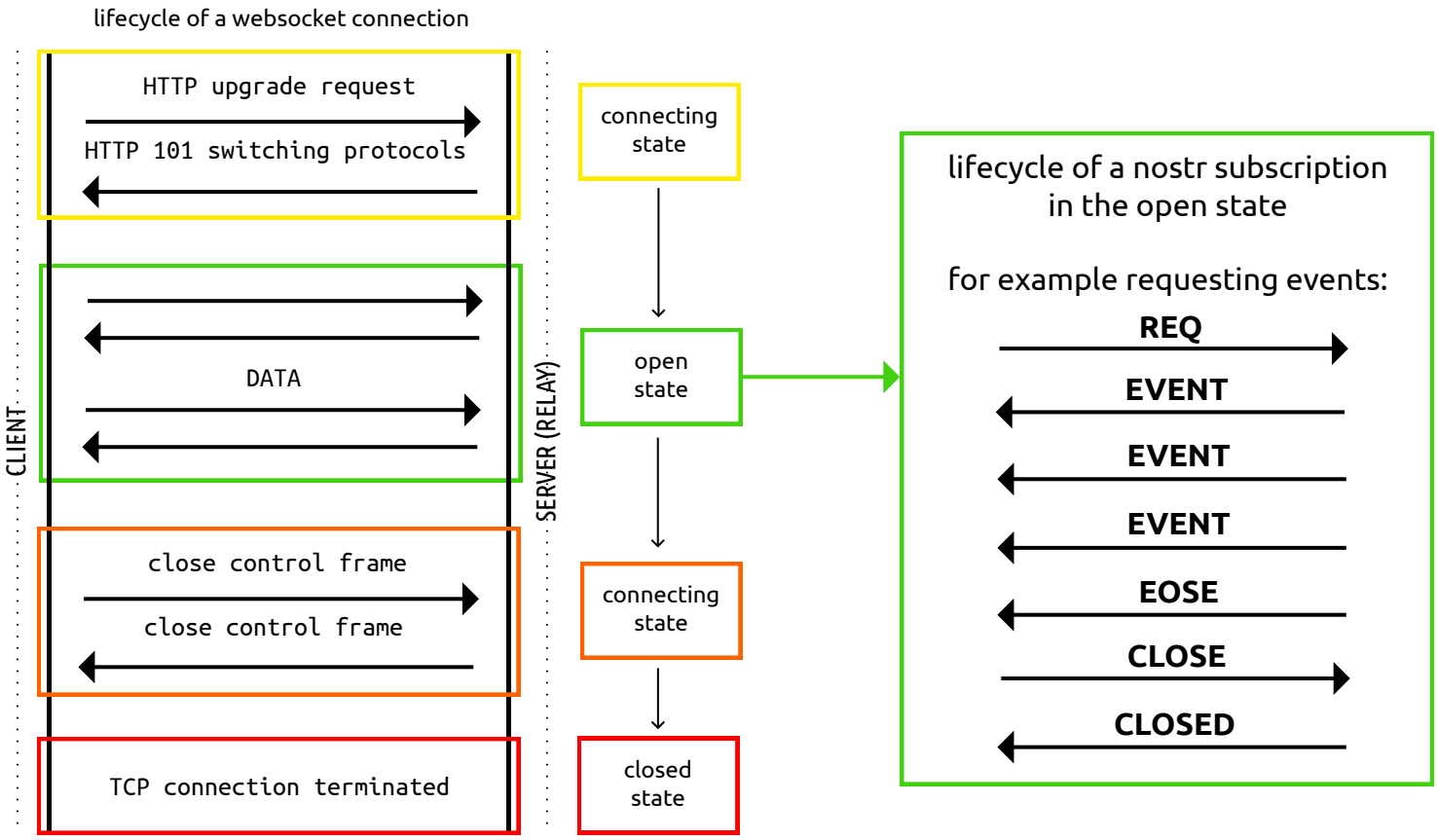
Communication between clients and relays

Relays expose a websocket endpoint to which clients can connect. Clients SHOULD open a single websocket connection to each relay and use it for all their subscriptions. Relays MAY limit number of connections from specific IP/client/etc.



From client to relay: sending events and creating subscriptions

Clients can send 3 types of messages, which must be JSON arrays, according to the following patterns:

- `["EVENT", <event JSON as defined above>]`, used to publish events.
- `["REQ", <subscription_id>, <filters1>, <filters2>, ...]`, used to request events and subscribe to new updates.
- `["CLOSE", <subscription_id>]`, used to stop previous subscriptions.

`<subscription_id>` is an arbitrary, non-empty string of max length 64 chars. It represents a subscription per connection. Relays MUST manage `<subscription_id>`s independently for each WebSocket connection. `<subscription_id>`s are not guaranteed to be globally unique.

`<filtersX>` is a JSON object that determines what events will be sent in that subscription, it can have the following attributes:

```
{
   "ids": <a list of event ids>,
   "authors": <a list of lowercase pubkeys, the pubkey of an event must be one of these>,
   "kinds": <a list of a kind numbers>,
   "#<single-letter (a-zA-Z)>": <a list of tag values, for #e — a list of event ids, for #p — a list of pubkeys, etc.>,
   "since": <an integer unix timestamp in seconds. Events must have a created_at >= to this to pass>,
   "until": <an integer unix timestamp in seconds. Events must have a created_at <= to this to pass>,
   "limit": <maximum number of events relays SHOULD return in the initial query>
}
```

Upon receiving a `REQ` message, the relay SHOULD return events that match the filter. Any new events it receives SHOULD be sent to that same websocket until the connection is closed, a `CLOSE` event is received with the same `<subscription_id>`, or a new `REQ` is sent using the same `<subscription_id>` (in which case a new subscription is created, replacing the old one). Filter attributes containing lists (`ids`, `authors`, `kinds` and `tag` filters like `#e`) are JSON arrays with one or more values. At least one of the arrays' values must match the relevant field in an event for the condition to be considered a match. For scalar event attributes such as `authors` and `kind`, the attribute from the event must be contained in the filter list. In the case of tag attributes such as `#e`, for which an event may have multiple values, the event and filter condition values must have at least one item in common.

The `ids`, `authors`, `#e` and `#p` filter lists MUST contain exact 64-character lowercase hex values.

The `since` and `until` properties can be used to specify the time range of events returned in the subscription. If a filter includes the `since` property, events with `created_at` greater than or equal to since are considered to match the filter. The until property is similar except that `created_at` must be less than or equal to `until`. In short, an event matches a filter if `since <= created_at <= until` holds.

All conditions of a filter that are specified must match for an event for it to pass the filter, i.e., multiple conditions are interpreted as `&&` conditions.

A `REQ` message may contain multiple filters. In this case, events that match any of the filters are to be returned, i.e., multiple filters are to be interpreted as `||` conditions.