# ZeroMQ

Paulo Sérgio Almeida

Grupo de Sistemas Distribuídos
Departamento de Informática
Universidade do Minho

# General socket semantics

- All sockets can connect or bind.
- Binding using TCP creates a listening socket at the node.
- Connect is asynchronous, returning immediately.
- If connection is broken, reconnection is attempted after a delay.

# Message semantics

- A message can be made of several parts (frames).
- Each part is a sequence of bytes.
- A message is delivered atomically: all parts or none.
- A message is delivered at most once to any peer.
- All messages between immediate peers are delivered in order.
- A message (single or multipart) must fit in memory.
- To send multipart message:
  - `socket.sendMore(msg)` to send all parts except final one,
  - `socket.send(msg)` for the final part.
- To receive:
  - `socket.recv()` dequeues one part,
  - `socket.hasReceiveMore()` to test if there are more parts,
  - when first `socket.recv()` returns, all parts are already at node.

# Valid socket type combinations

| | REQ | REP | DEALER | ROUTER | PUB | XPUB | SUB | XSUB | PUSH | PULL | PAIR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| REQ | | ✓ | | ✓ | | | | | | | |
| REP | ✓ | | ✓ | | | | | | | | |
| DEALER | | ✓ | ✓ | ✓ | | | | | | | |
| ROUTER | ✓ | | ✓ | ✓ | | | | | | | |
| PUB | | | | | | | ✓ | ✓ | | | |
| XPUB | | | | | | | ✓ | ✓ | | | |
| SUB | | | | | ✓ | ✓ | | | | | |
| XSUB | | | | | ✓ | ✓ | | | | | |
| PUSH | | | | | | | | | | ✓ | |
| PULL | | | | | | | | | ✓ | | |
| PAIR | | | | | | | | | | | ✓ |

# Identities and Addresses

- A message is composed by:
  - zero or more address frames, each containing an identity;
  - an empty delimiter frame;
  - zero or more data frames;
- Simpler socket types only let application see data frames.
- Others let full message go through to application.
- Some socket types prepend or remove address frames.
- Identities:
  - may be choosen automatically by ROUTER sockets;
  - may be created explicitly by application; assigned by `socket.setIdentity` so that a REQ, DEALER, or ROUTER connecting to a ROUTER announces it.

# REQ socket type

- REQ socket type acts as client for simple request-reply patterns.
- May be connected to several REP or ROUTER peers.
- Sends requests and receives replies in alternation.
- For outgoing messages:
  - Prefixes outgoing messages with an empty delimiter frame.
  - Sends outgoing messages to connected peers in round-robin.
  - Blocks on sending or returns error when it has no connected peers.
- For incoming messages:
  - Accepts incoming message from the peer of the pending request.
  - Removes starting empty frame, delivering remaining data frames.
  - Discards silently messages from other peers.

# REP socket type

- REP socket type acts as server for simple request-reply patterns.
- May be connected to several REQ or DEALER peers.
- Receives request and sends back reply in alternation.
- For incoming messages:
  - Receives incoming messages using fair-queuing.
  - Removes and keeps address frames and empty delimiter frame.
  - Pass remaining data frames to application.
- For outgoing messages:
  - Waits for reply message from application.
  - Prefixes message with address frames and delimiter that it kept.
  - Sends prefixed message back to requesting peer, without blocking.
  - Discards reply or returns error if that peer no longer connected.

# DEALER socket type

- May be connected to several REP, DEALER or ROUTER peers.
- Maintains independent incoming and outgoing queue per peer.
- When connecting,creates and maintains queues.
- When binding, discards queues and messages upon disconnect.
- For outgoing messages:
  - Considers peer available if it has a non-full outgoing queue.
  - Sends outgoing messages to available peers in round-robin.
  - Blocks on sending or returns error when it has no available peers.
- For incoming messages:
  - Receives incoming messages from peers using fair-queuing.
  - Delivers full messages unmodified to application.

# ROUTER socket type

- May be connected to several REQ, DEALER or ROUTER peers.
- Maintains independent incoming and outgoing queue per peer.
- When connecting, creates and maintains queues.
- When binding, discards queues and messages upon disconnect.
- Identifies each peer using a locally generated unique identity.
- Allows peers to specify identity explicitly through `setIdentity`.
- For incoming messages:
  - Receives incoming messages using fair-queuing.
  - Prefixes each message with an identity frame according to peer.
- For outgoing messages:
  - Removes first message frame and uses it as peer identity.
  - Routes message to corresponding outgoing queue if it is not full.
  - Silently drops message or returns error if queue missing or full.

# PUB socket type

- May be connected to several SUB or XSUB peers.
- Maintains single outgoing queue for each connected subscriber.
- When connecting, creates and maintains queue.
- When binding, discards queue and messages upon disconnect.
- For outgoing messages:
  - Comparises subscriptions against start of first message frame.
  - Enqueues message to queue if it matches subscription.
  - Silently drops message if queue for a subscriber is full.
- For incoming messages:
  - Process subscribe and unsubscribe requests.
  - Need *n* unsubscribes to cancel *n* subscribes.
  - Does not deliver messages to application.

# XPUB socket type

- May be connected to several SUB or XSUB peers.
- Exposes subscription messages to application.
- Maintains independent incoming and outgoing queue per peer.
- When connecting, creates and maintains queues.
- When binding, discards queues and messages upon disconnect.
- For outgoing messages:
  - Comparises subscriptions against start of first message frame.
  - Enqueues message to queue if it matches subscription.
  - Silently drops message if queue for a subscriber is full.
- For incoming messages:
  - Process subscribe and unsubscribe requests.
  - Need *n* unsubscribes to cancel *n* subscribes.
  - Deliver subscription messages to application if impacting upstream.

# SUB socket type

- May be connected to several PUB or XPUB peers.
- Maintains single incoming queue for each connected publisher.
- When connecting, creates and maintains queue.
- When binding, discards queue and messages upon disconnect.
- For incoming messages:
  - Silently discard messages if queue for a publisher is full.
  - Receive incoming messages from publishers using fair-queuing.
  - May filter messages according to subscriptions, using prefix match.
- For processing subscriptions:
  - Sends subscribe and unsubscribe messages to publishers.

# XSUB socket type

- May be connected to several PUB or XPUB peers.
- Exposes subscription messages to application.
- Maintains independent incoming and outgoing queue per peer.
- When connecting,creates and maintains queues.
- When binding, discards queues and messages upon disconnect.
- For incoming messages:
  - Silently discard messages if queue for a publisher is full.
  - Delivers incoming messages from publishers using fair-queuing.
  - May filter messages according to subscriptions, using prefix match.
- For outgoing messages:
  - Sends messages to all connected publishers.
  - Silently drops message if outgoing queue for a publisher is full.
- For processing subscriptions:
  - Sends subscribe and unsubscribe messages to publishers.

# PUSH socket type

- May be connected to several PULL peers.
- Maintains single outgoing queue per connected peer.
- When connecting, creates and maintains queue.
- When binding, discards queue and messages upon disconnect.
- For outgoing messages:
  - Consider peer available when it has a non-full outgoing queue.
  - Sends outgoing messages to connected peers in round-robin.
  - Blocks on sending or returns error when it has no available peers.

# PULL socket type

- May be connected to several PUSH peers.
- Maintains single incoming queue per connected peer.
- When connecting, creates and maintains queue.
- When binding, discards queue and messages upon disconnect.
- For incoming messages:
  - Receives incoming messages from peers using fair-queuing.
  - Delivers messages unmodified to application.