SECURE ZMQ

Securing ZMQ using GNUTLS

AGENDA

- ZMQ Overview
- GNUTLS Overview
- Secure ZMQ using GNUTLS
- DEMO

Introduction to ØMQ

ZeroMQ - The Intelligent Transport Layer

ØMQ in a nutshell

- Carries messages across inproc, IPC, TCP and multicast.
- Acts as a concurrency framework.
- Connect N-to-N via fanout, pubsub, pipeline, request-reply.
- 30+ languages including C, C++, Java, .NET, Python.
- Most OSes including Linux, Windows, OS X.
- LGPL licensed free software.

Simple Client-Server Example

```
void *context = zmq_ctx_new ();
void *requester = zmq_socket (context, ZMQ_REQ);
zmq_connect (requester, "tcp://localhost:5555");
```

```
//  Socket to talk to clients
void *context = zmq_ctx_new ();
void *responder = zmq_socket (context, ZMQ_REP);
int rc = zmq_bind (responder, "tcp://*:5555");
assert (rc == 0);
```

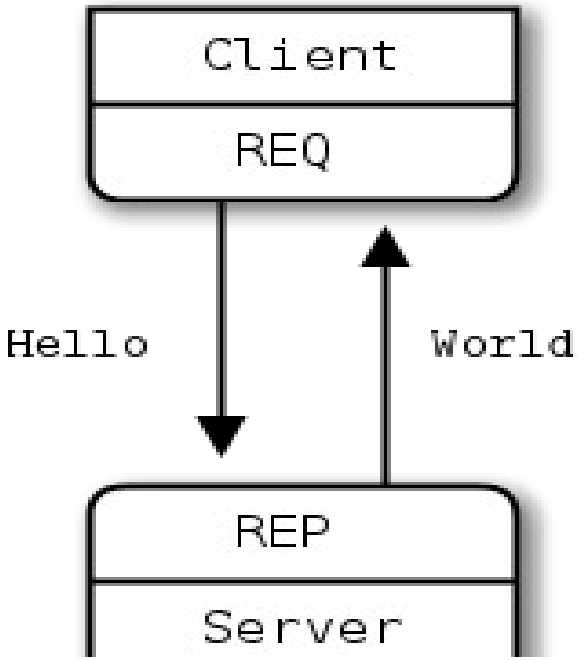
- Create a context
- Create a new socket of the desired type
- Connect(create outgoing connection) or bind(for accepting incoming connections) the socket to an endpoint.
- Use zmq-send() and zmq_recv() to send and receive messages.

Basic Messaging patterns defined by ZMQ

- REQ-REP
- ROUTER-DEALER
- PUB-SUB
- PUSH-PULL
- Exclusive pair

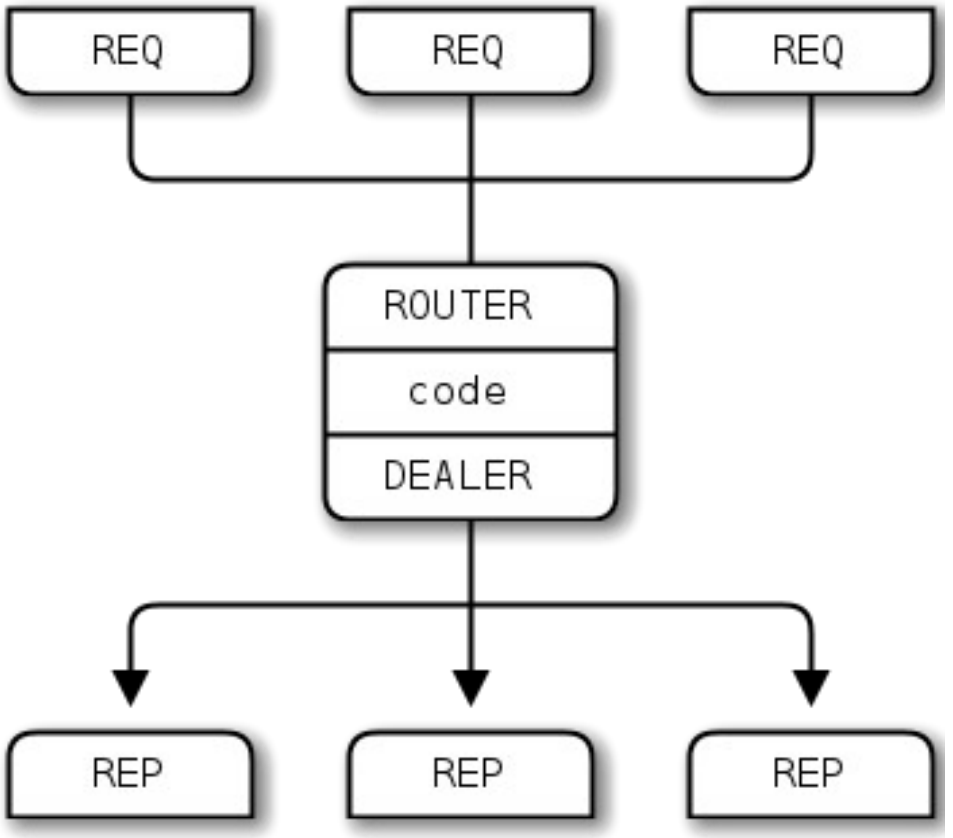
REQ-REP

- used for sending requests from a ZMQ_REQ client to one or more ZMQ_REP services and receiving subsequent replies.



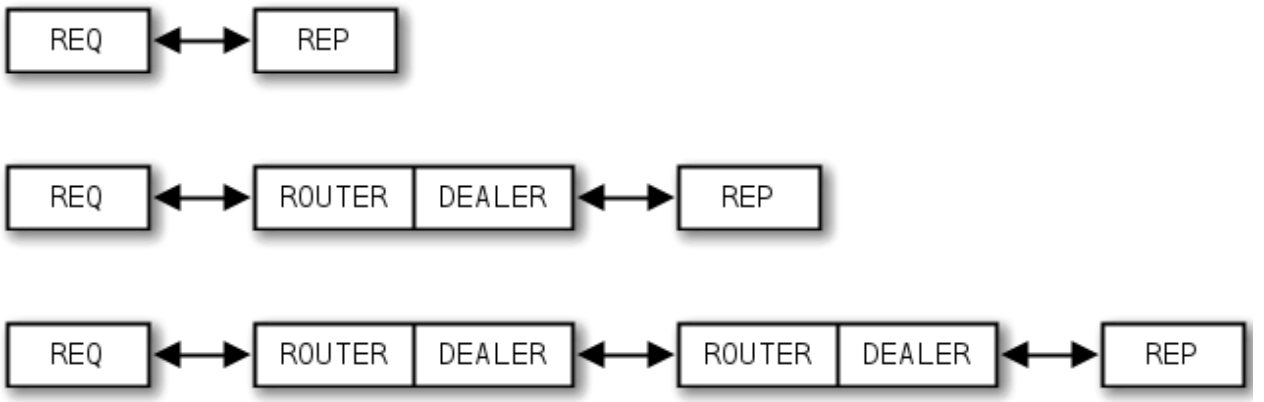
ROUTER-DEALER

- an advanced pattern that can be used for extending req-rep patterns. Allows asynchronous message exchange



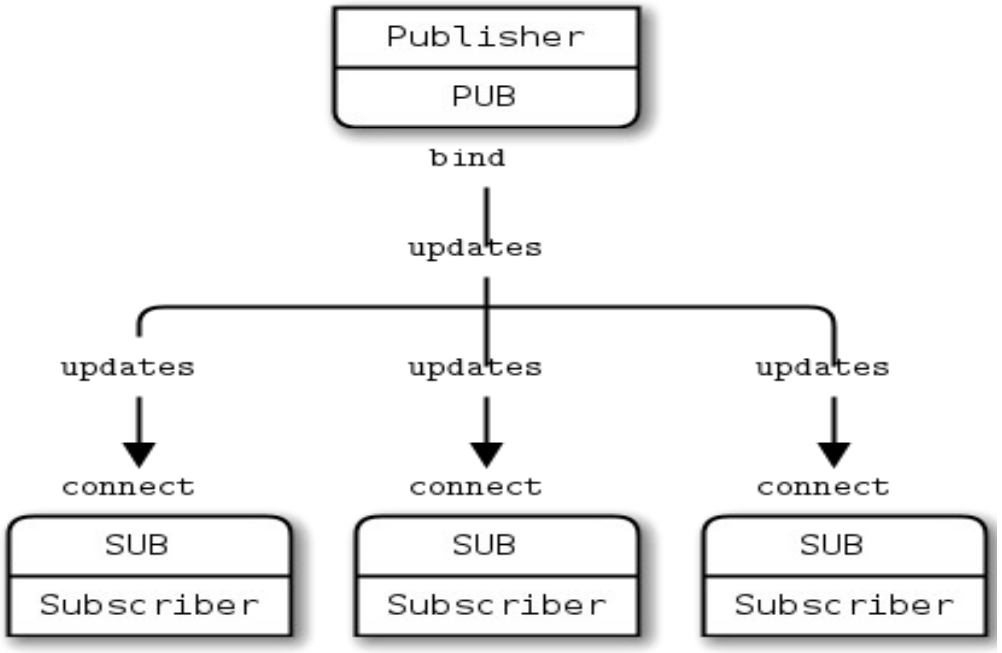
ROUTER DEALER PROXY

(extending request-reply)



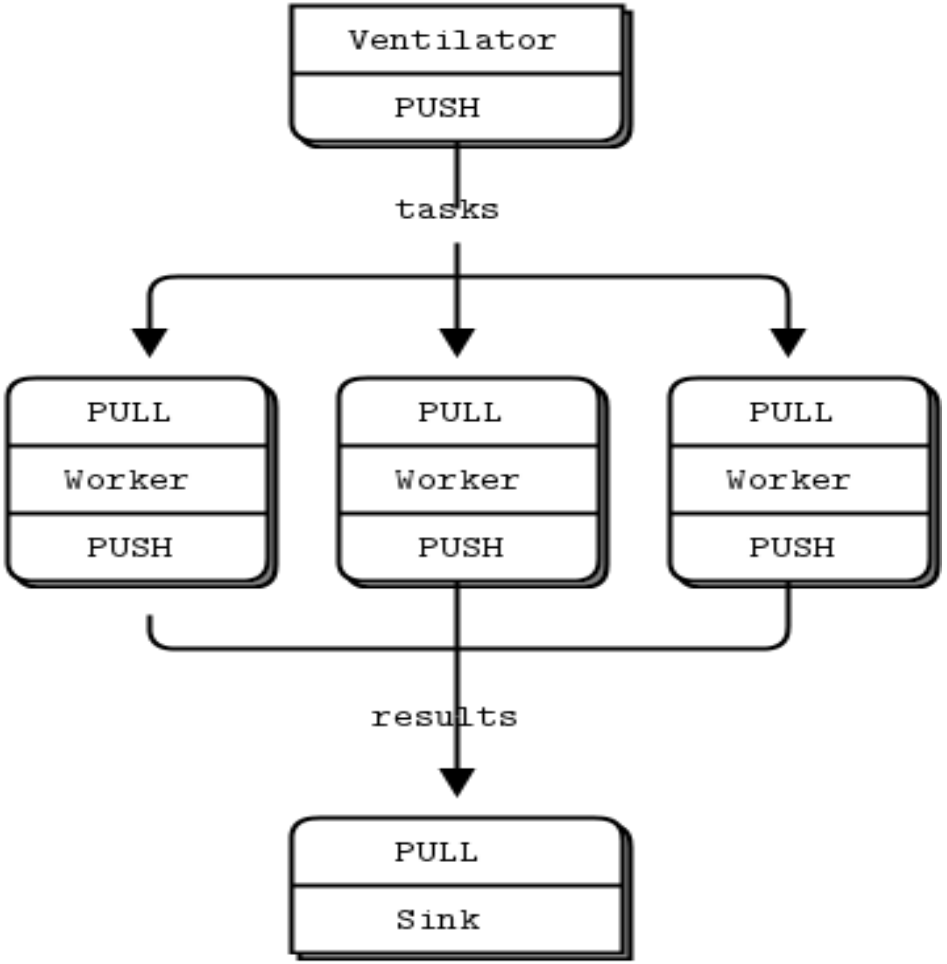
PUB-SUB

- used for one-to-many distribution of data from a single publisher to multiple subscribers in a fan out fashion.



PUSH-PULL

- used for distributing data to nodes arranged in a pipeline. Data always flows down the pipeline, and each stage of the pipeline is connected to at least one node.



Exclusive Pair

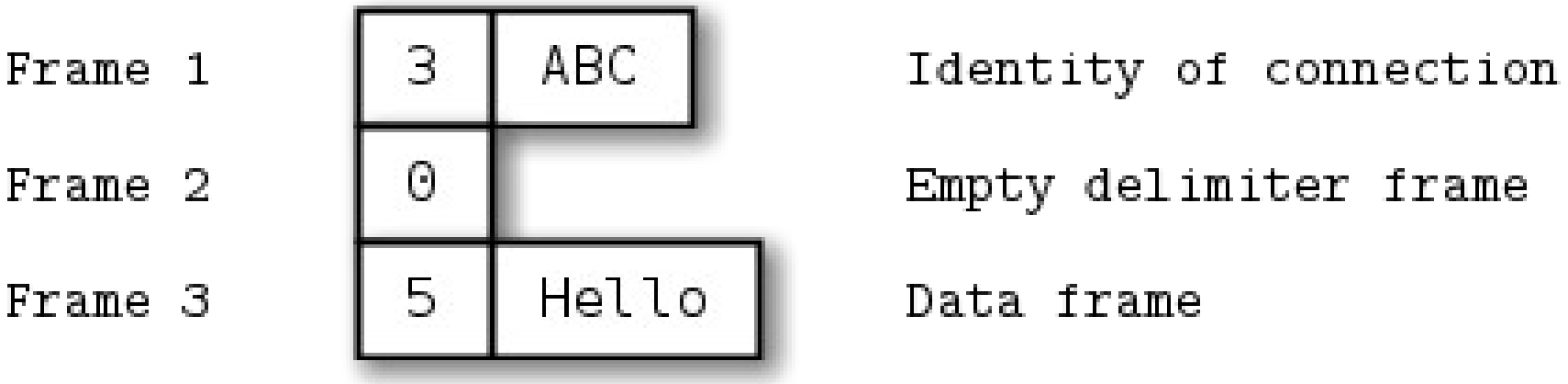
- used to connect a peer to precisely one other peer.
- can be used for inter-thread communication across the inproc transport.

Summary of ZMQ_PAIR characteristics	
Compatible peer sockets	*ZMQ_PAIR*
Direction	Bidirectional
Send/receive pattern	Unrestricted
Incoming routing strategy	N/A
Outgoing routing strategy	N/A
Action in mute state	Block

MULTITHREADING USING ZMQ

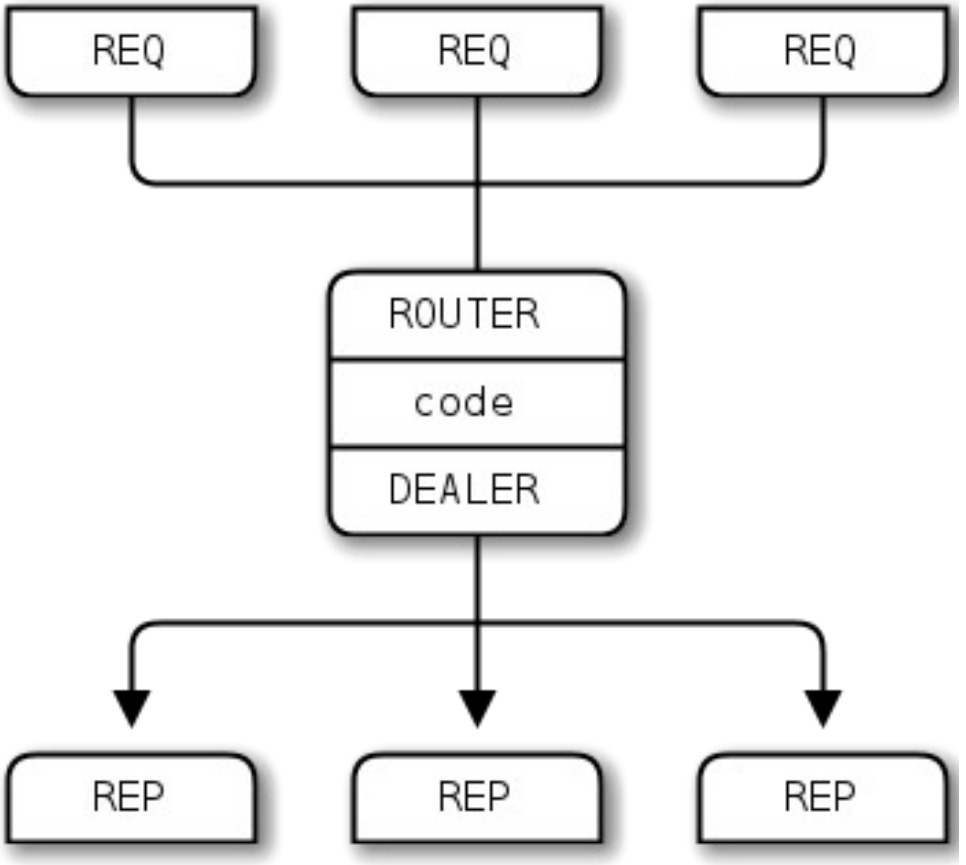
- No mutexes, locks required
- Inter thread communication using pair sockets under a shared context
- ZMQ sockets are not thread safe

MESSAGE FRAMES



ROUTER DEALER PROXY

(special use case)



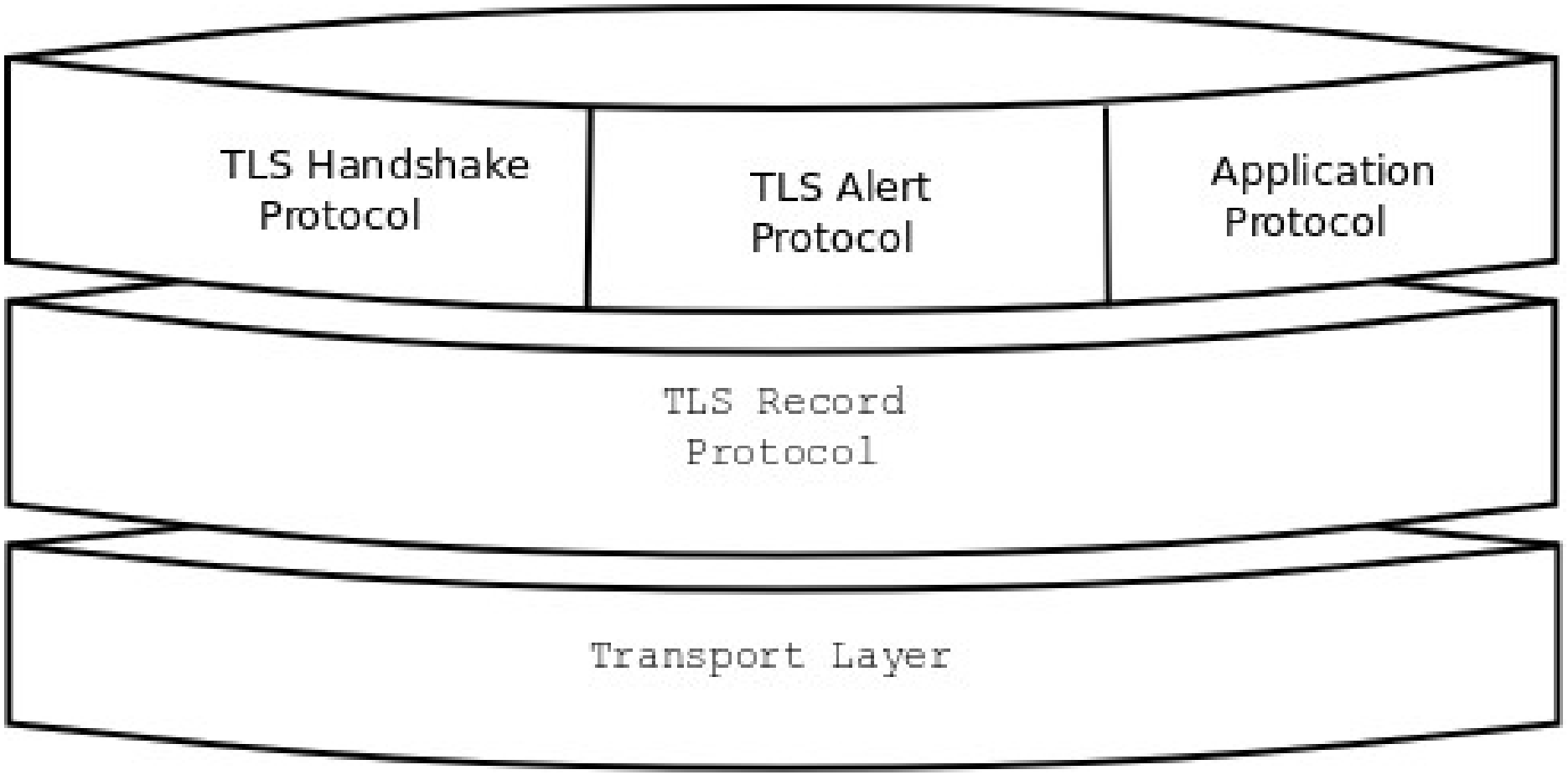
GNUTLS

The GNU Transport Layer Security Library

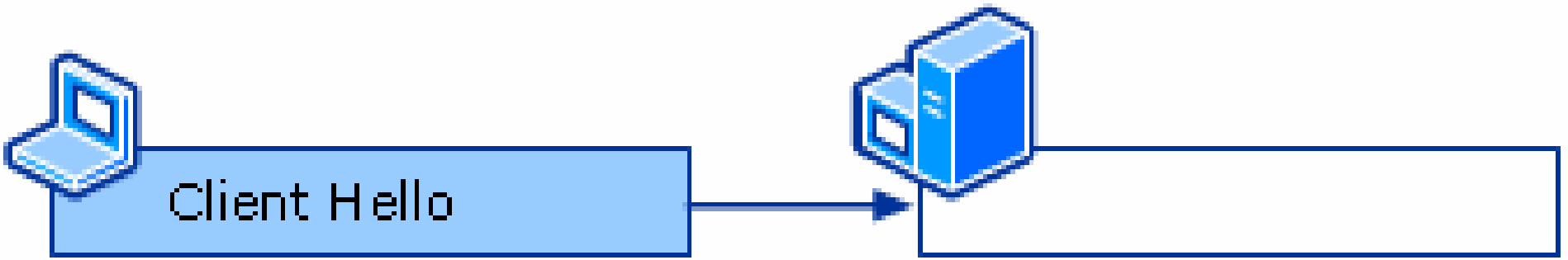
GNUTLS

- Implements TLS, SSL and DTLS protocols
- Supports SRP, PSK, certificate (x.509 / OpenPGP) and Anonymous authentication
- LGPL licensed

GNUTLS LAYERS

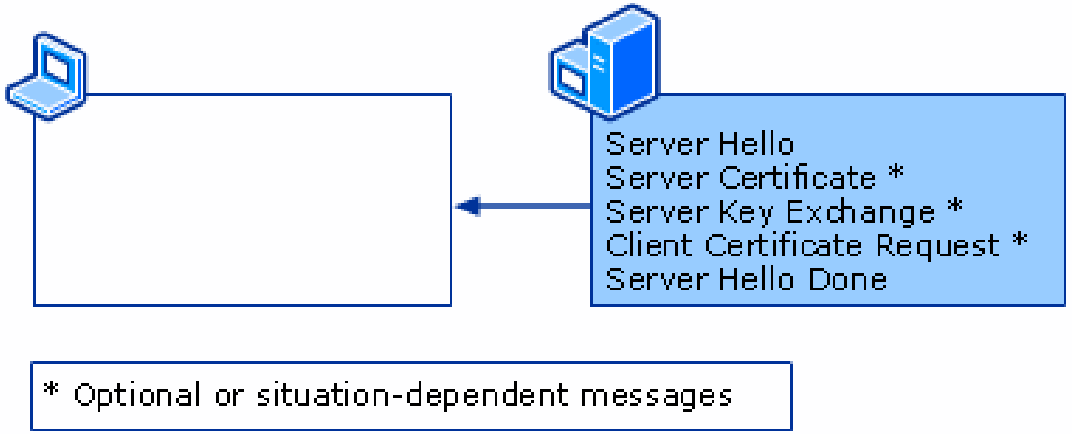


TLS HANDSHAKE



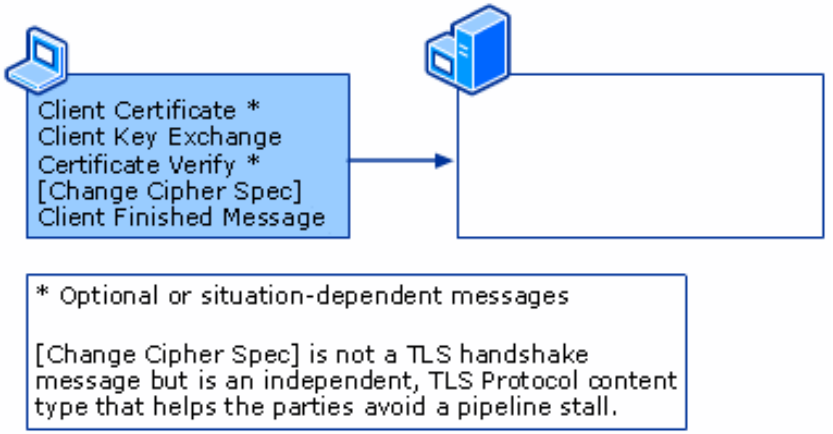
- Version number
- Cipher suite
- Client Random
- Session Identification (optional)

TLS HANDSHAKE



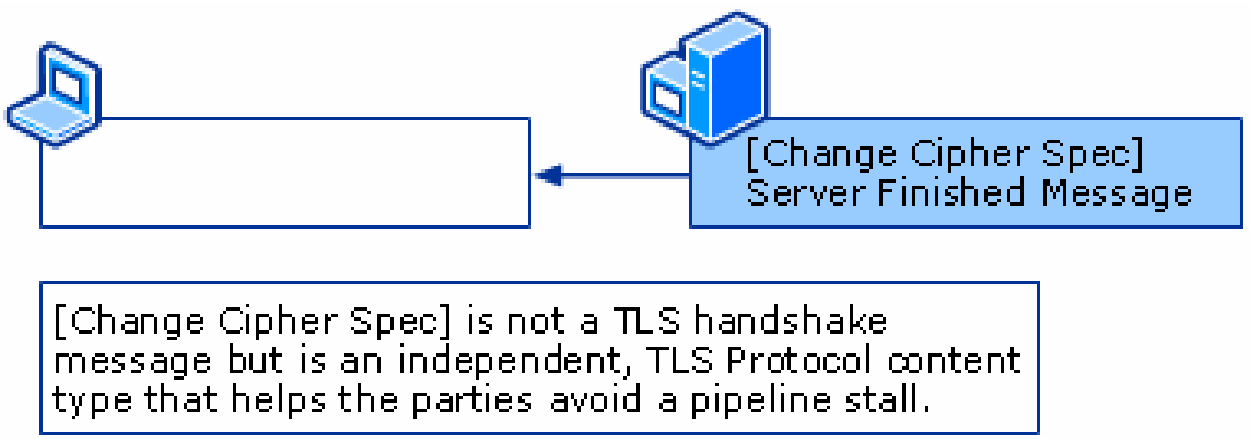
- Version
- Server random
- Cipher
- Server Certificate
- Client certificate request

TLS HANDSHAKE



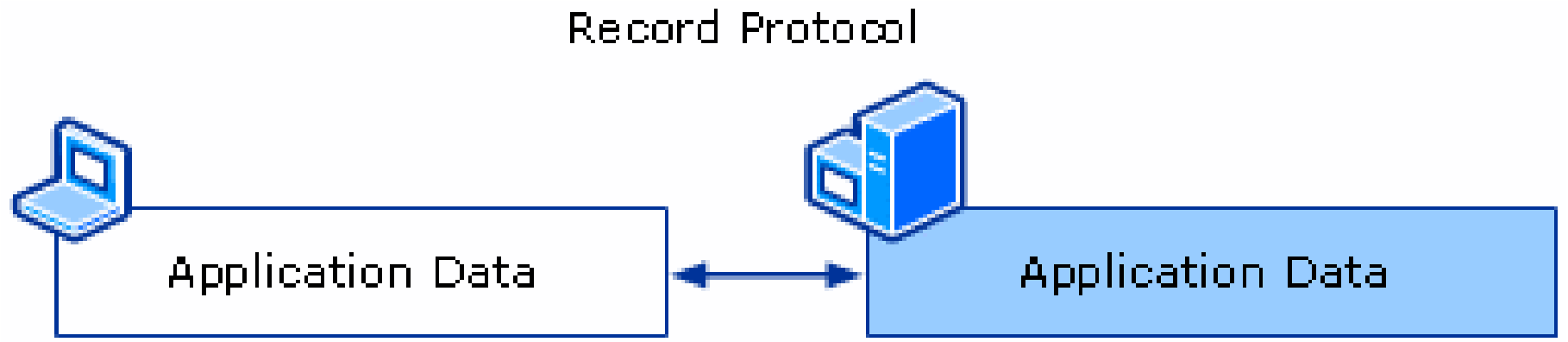
- Client Certificate
- Pre-master secret
- Change Cipher Spec
- Client Hello Done

TLS HANDSHAKE



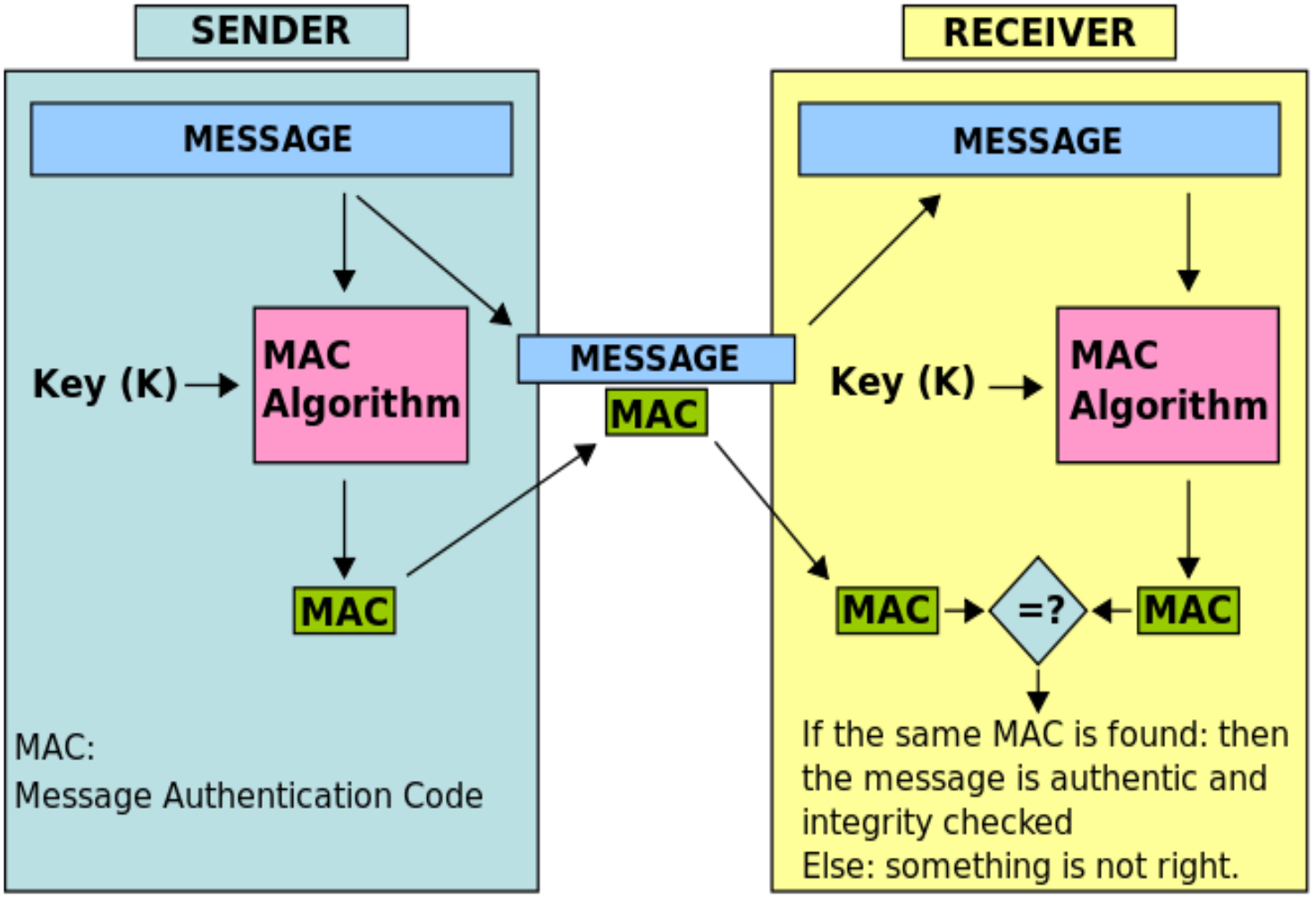
- Change Cipher Spec
- Server Hello Done

RECORD PROTOCOL



- Symmetric key encryption using Master secret
- MAC / HMAC for integrity and authenticity check

MESSAGE AUTHENTICATION



PERFECT FORWARD SECRECY

- Private key alone is not sufficient to decrypt past messages
- Pre-master secret not sent over the wire
- DHE, ECDHE
- Significant performance overhead

USING GNUTLS

- gnutls_global_init()
- Set cipher priority
- Associate credentials
- Initialize session
- Set up transport layer

USING GNUTLS

Transport Settings

- Create socket
- Associate socket with session
- Set push/pull callback functions
- Set timeout function

SESSION RESUMPTION

- Server side:- Database

  -store session data in a server side database.

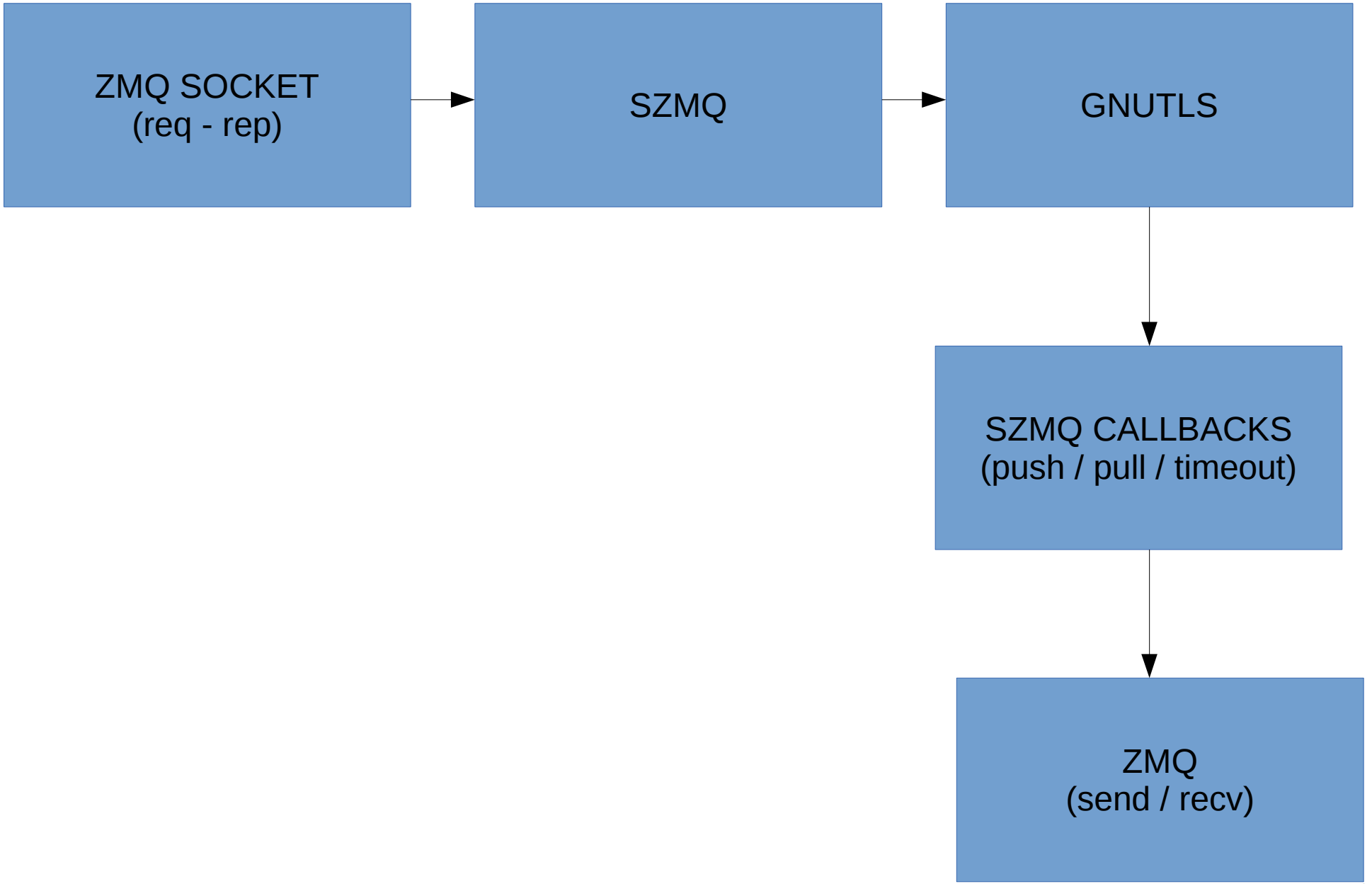
- Client Side:- Session tickets

  -session data stored on the client side.

SZMQ: SECURE ZMQ

TLS over ZMQ

SZMQ WORKFLOW



USING SZMQ

- szmq_global_init ()
- Supply credentials
- Set cipher priority*
- Create ZMQ context and socket
- szmq_session_init (socket)
- Set ZMQ flags*
- send / recv
- szmq_bye ()
- szmq_session_deinit ()
- szmq_global_deinit ()

** optional*

GNUTLS API

All GNUTLS API functions can be used in addition to SZMQ

- szmq_session.gnutls_session : GNUTLS session structure
- szmq_context.credentials : GNUTLS credentials
- szmq_context.priority : Cipher priority
- szmq_context.dh_params : Diffie-Hellman parameters

DIFFICULTIES FACED

- zmq_recv() reads entire buffer even if smaller size is specified, gnutls reads the buffer in parts.

  Used intermediate buffer.

- consecutive send / recv required for handshake, not supported by req-rep sockets.

  Used multipart messages.

- Failed handshake may cause socket to hang.

  Reset socket state, discard messages

- Push / pull callback functions required by gnutls can't have a flag argument.

  Created separate szmq_set_flag() function

EXTRA TECHNICAL DETAILS

The following slides contain detailed information about mentioned explained before.

REQ-REP

Summary of ZMQ_REQ characteristics	
Compatible peer sockets	*ZMQ_REP*, *ZMQ_ROUTER*
Direction	Bidirectional
Send/receive pattern	Send, Receive, Send, Receive, ...
Outgoing routing strategy	Round-robin
Incoming routing strategy	Last peer
Action in mute state	Block

Summary of ZMQ_REP characteristics	
Compatible peer sockets	*ZMQ_REQ*, *ZMQ_DEALER*
Direction	Bidirectional
Send/receive pattern	Receive, Send, Receive, Send, ...
Incoming routing strategy	Fair-queued
Outgoing routing strategy	Last peer
Action in mute state	Drop

ROUTER-DEALER

Summary of ZMQ_ROUTER characteristics	
Compatible peer sockets	*ZMQ_DEALER*, *ZMQ_REQ*, *ZMQ_ROUTER*
Direction	Bidirectional
Send/receive pattern	Unrestricted
Outgoing routing strategy	See text
Incoming routing strategy	Fair-queued
Action in mute state	Drop

Summary of ZMQ_DEALER characteristics	
Compatible peer sockets	*ZMQ_ROUTER*, *ZMQ_REP*, *ZMQ_DEALER*
Direction	Bidirectional
Send/receive pattern	Unrestricted
Outgoing routing strategy	Round-robin
Incoming routing strategy	Fair-queued
Action in mute state	Block

PUB-SUB

Summary of ZMQ_PUB characteristics	
Compatible peer sockets	*ZMQ_SUB*, *ZMQ_XSUB*
Direction	Unidirectional
Send/receive pattern	Send only
Incoming routing strategy	N/A
Outgoing routing strategy	Fan out
Action in mute state	Drop

Summary of ZMQ_SUB characteristics	
Compatible peer sockets	*ZMQ_PUB*, *ZMQ_XPUB*
Direction	Unidirectional
Send/receive pattern	Receive only
Incoming routing strategy	Fair-queued
Outgoing routing strategy	N/A
Action in mute state	Drop

PUSH-PULL

Summary of ZMQ_PUSH characteristics	
Compatible peer sockets	*ZMQ_PULL*
Direction	Unidirectional
Send/receive pattern	Send only
Incoming routing strategy	N/A
Outgoing routing strategy	Round-robin
Action in mute state	Block

Summary of ZMQ_PULL characteristics	
Compatible peer sockets	*ZMQ_PUSH*
Direction	Unidirectional
Send/receive pattern	Receive only
Incoming routing strategy	Fair-queued
Outgoing routing strategy	N/A
Action in mute state	Block

Exclusive Pair

Summary of ZMQ_PAIR characteristics	
Compatible peer sockets	*ZMQ_PAIR*
Direction	Bidirectional
Send/receive pattern	Unrestricted
Incoming routing strategy	N/A
Outgoing routing strategy	N/A
Action in mute state	Block

GNUTLS Details
PERFECT FORWARD SECRECY

Benchmarks

Key exchange	Parameters	Transactions/sec
DHE-RSA	1024-bit RSA key, 1024-bit DH parameters	345.53
ECDHE-RSA	1024-bit RSA key, 192-bit ECDH parameters	604.92
ECDHE-ECDSA	192-bit ECDSA key, 192-bit ECDH parameters	595.84
RSA	1024-bit RSA key	994.59

GNUTLS Details
PERFECT FORWARD SECRECY

Benchmarks

Key exchange	Parameters	Transactions/sec
DHE-RSA	1776-bit RSA key, 1776-bit DH parameters	98.26
ECDHE-RSA	1776-bit RSA key, 192-bit ECDH parameters	352.41
ECDHE-ECDSA	192-bit ECDSA key, 192-bit ECDH parameters	595.84
RSA	1776-bit RSA key	460.08

GNUTLS references

- GNUTLS manual: http://www.gnutls.org/manual/gnutls.html
- http://technet.microsoft.com/en-us/library/cc783349%28v=WS.10%29.aspx
- http://nmav.gnutls.org/2011/12/price-to-pay-for-perfect-forward.html

ZMQ references

- http://www.zeromq.org/
- http://api.zeromq.org/

THANK YOU!