Saint Petersburg
3-4 November 2017

Asyncio & SMTP

Konstantin Volkov

Own SMTP server – why?

- Cloud Email Security service
- SLA 99.999%
- 1M+ users over the world
- Corporate instructions and practices
- Agile configuration
- Internal services integration
- Guaranteed quality

Piter Py #4

SMTP Python history

- 1965 - First email

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0
- 1997 - smtplib

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0
- 1997 - smtplib
- 2000 - Python 2.0

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0
- 1997 - smtplib
- 2000 - Python 2.0
- 2001 - smtpd.py

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0
- 1997 - smtplib
- 2000 - Python 2.0
- 2001 - smtpd.py
- 2008 - ESMTP (extended SMTP)

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0
- 1997 - smtplib
- 2000 - Python 2.0
- 2001 - smtpd.py
- 2008 - ESMTP (extended SMTP)
- 2014 - aiosmtpd, aiosmtplib

Piter Py #4

SMTP Python history

- 1965 - First email
- 1982 - SMTP (Simple Mail Transfer Protocol)
- 1994 - Python 1.0
- 1997 - smtplib
- 2000 - Python 2.0
- 2001 - smtpd.py
- 2008 - ESMTP (extended SMTP)
- 2014 - aiosmtpd, aiosmtplib
- 2017 - you are here

SMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1

SMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
HELO mailout1.phrednet.com
> 250 pleased to meet you

SMTP

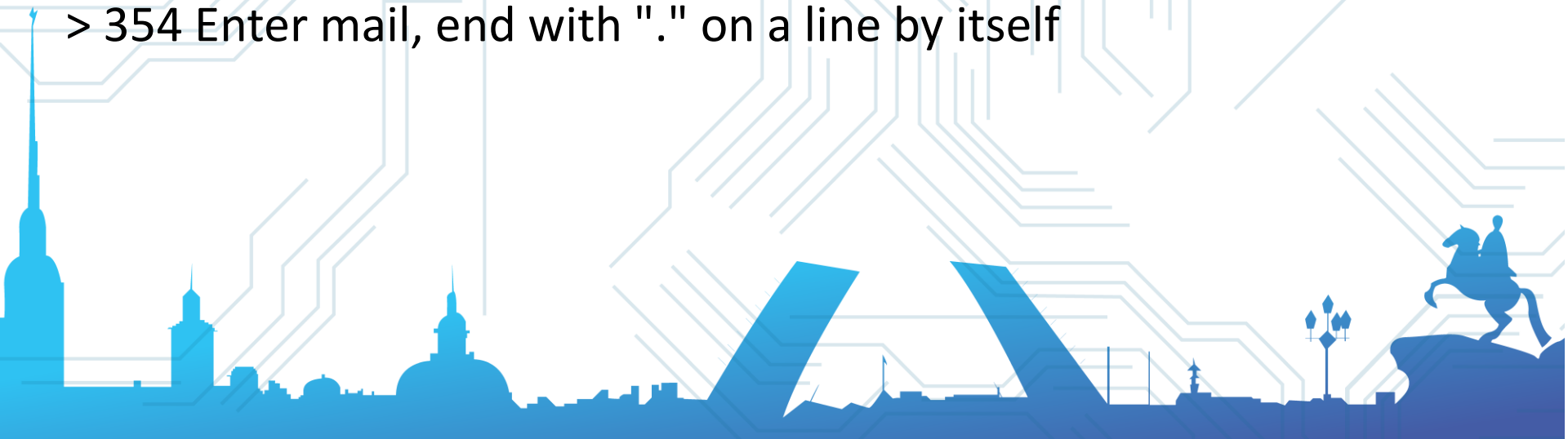
> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
HELO mailout1.phrednet.com
> 250 pleased to meet you
MAIL FROM:<xxxx@example.com>
> 250 Sender ok

SMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
HELO mailout1.phrednet.com
> 250 pleased to meet you
MAIL FROM:<xxxx@example.com>
> 250 Sender ok
RCPT TO:<yyyy@example.com>
> 250 Recipient ok

SMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
HELO mailout1.phrednet.com
> 250 pleased to meet you
MAIL FROM:<xxxx@example.com>
> 250 Sender ok
RCPT TO:<yyyy@example.com>
> 250 Recipient ok
DATA
> 354 Enter mail, end with "." on a line by itself



SMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
HELO mailout1.phrednet.com
> 250 pleased to meet you
MAIL FROM:<xxxx@example.com>
> 250 Sender ok
RCPT TO:<yyyy@example.com>
> 250 Recipient ok
DATA
> 354 Enter mail, end with "." on a line by itself
This is test message.\r\n.\r\n
> 250 Message accepted for delivery

ESMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
EHLO mailout1.phrednet.com

ESMTP

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
EHLO mailout1.phrednet.com
> 250-smtpserver
> 250-AUTH LOGIN PLAIN
> 250-8BITMIME
> 250-STARTTLS
> 250 PIPELINING

STARTTLS

> 220 example.org ESMTP Sendmail 8.13.1/8.13.1
EHLO mailout1.phrednet.com
> 250-smtpserver
> 250-AUTH LOGIN PLAIN
> 250-8BITMIME
> 250-STARTTLS
> 250 PIPELINING
STARTTLS
> 220 Ready to start TLS

smtplib

- 20 years old
- Supports ESMTP
- Supports SSL/STARTTLS
- Direct socket handling

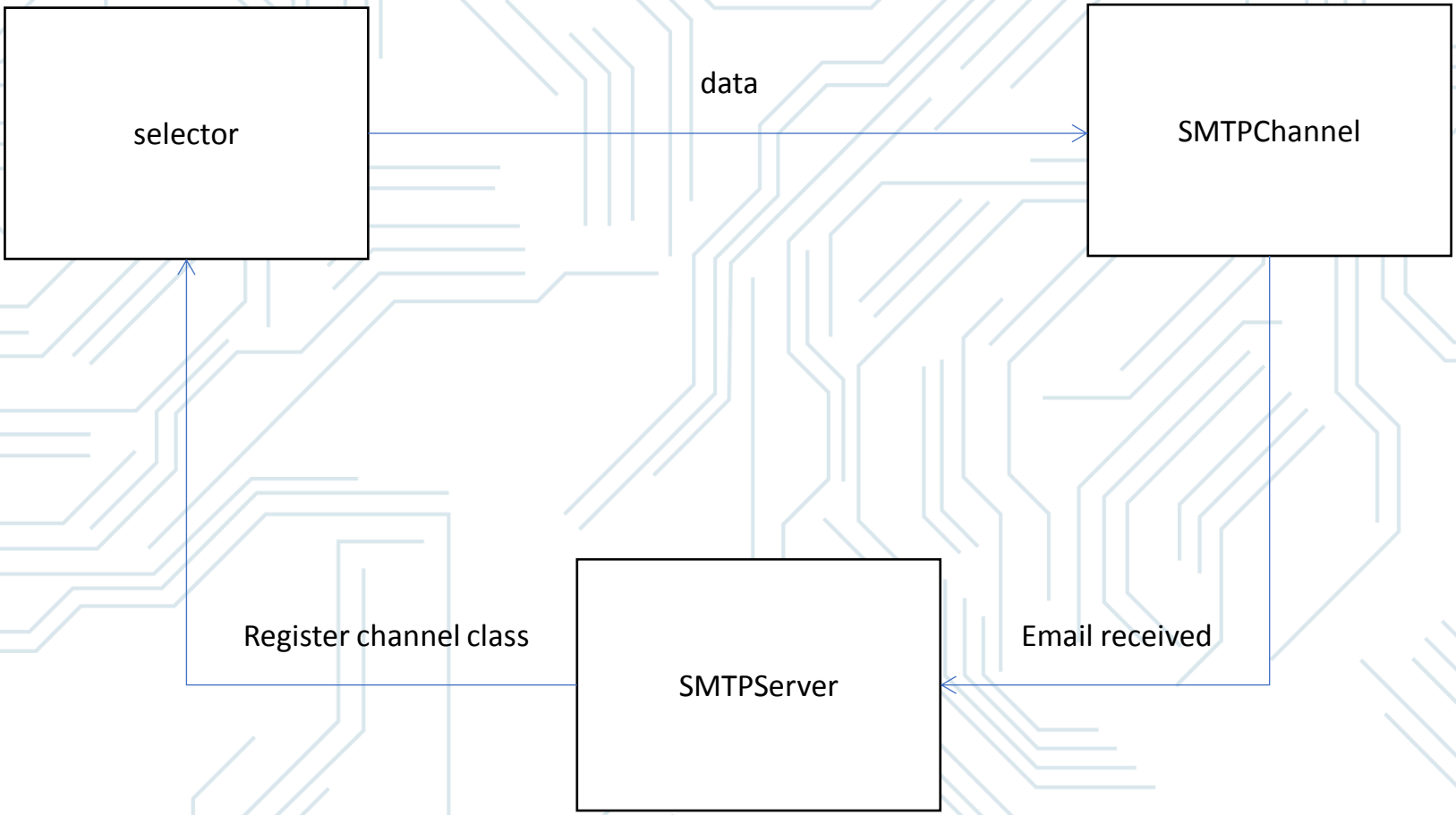
smtplib

```
if self.file is None:
    self.file = self.sock.makefile('rb')
while 1:
    try:
        line = self.file.readline(_MAXLINE + 1)
    except OSError as e:
        self.close()
        raise SMTPServerDisconnected("Connection unexpectedly closed: "
                                     + str(e))
```

smtpd

- 17 years old
- Based on asynchat/asyncore
- Written by Barry Warsaw
- Basic ESMTP support
- No encryption

smtpd


selector
data
SMTPChannel
Register channel class
SMTPServer
Email received

smtpd

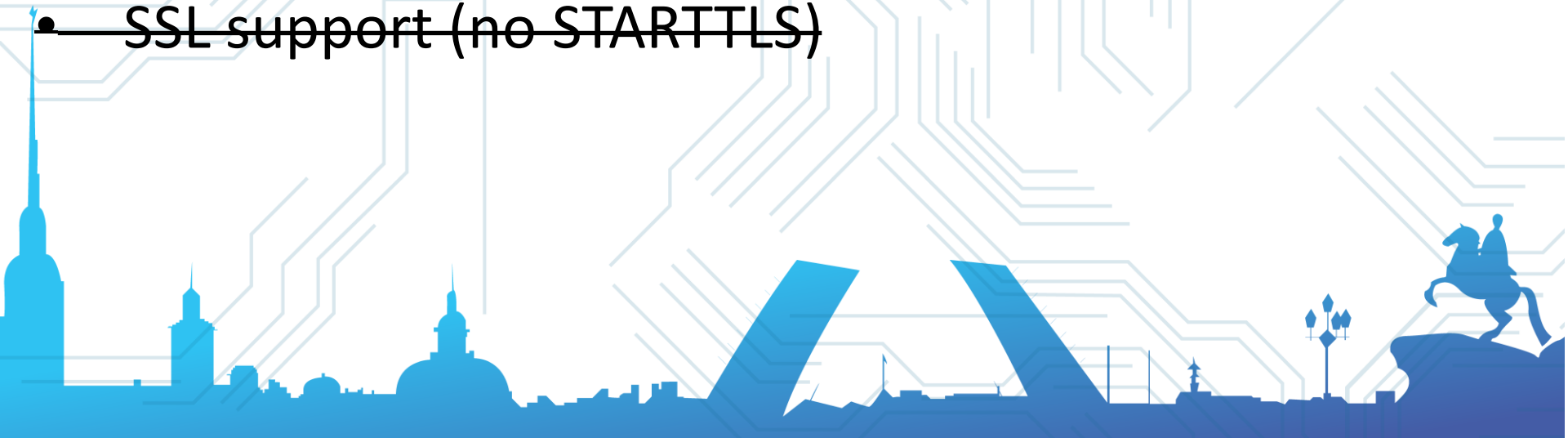
```
# Implementation of base class abstract method
def found_terminator(self):
    line = self._emptystring.join(self.received_lines)
    print('Data:', repr(line), file=DEBUGSTREAM)
    self.received_lines = []
    if self.smtp_state == self.COMMAND:...
    else:
        if self.smtp_state != self.DATA:
            self.push('451 Internal confusion')
            self.num_bytes = 0
            return
        if self.data_size_limit and self.num_bytes > self.data_size_limit:
            self.push('552 Error: Too much mail data')
            self.num_bytes = 0
            return
        # Remove extraneous carriage returns and de-transparency according
        # to RFC 5321, Section 4.5.2.
        data = []
        for text in line.split(self._linesep):
```

aiosmtpd

- 3 years old (new)
- Based on asyncio
- Written by Barry Warsaw to replace stmtpd
- Basic ESMTP support
- SSL support (no STARTTLS)

aiosmtpd

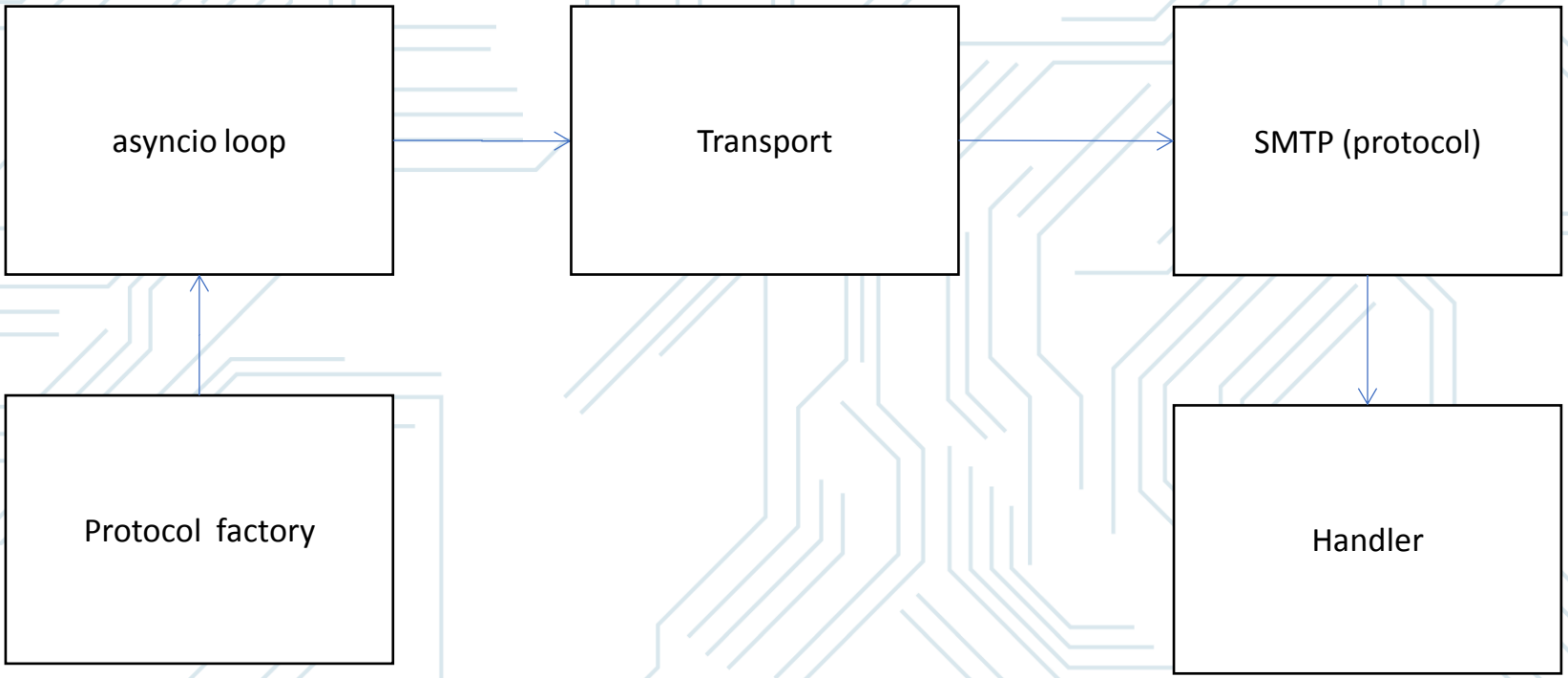
- 3 years old (new)
- Based on asyncio
- Written by Barry Warsaw to replace stmtpd
- ~~Basic ESMTP support~~
- ~~SSL support (no STARTTLS)~~



aiosmtpd

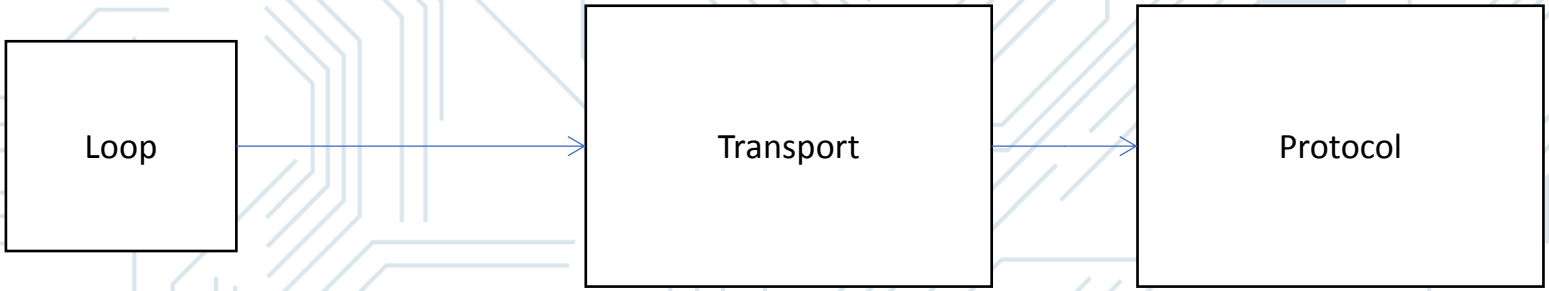
- 3 years old (new)
- Based on asyncio
- Written by Barry Warsaw to replace smtpd
- Full ESMTP support
- SSL support (+ STARTTLS)

aiosmtpd


asyncio loop
Transport
SMTP (protocol)
Protocol factory
Handler

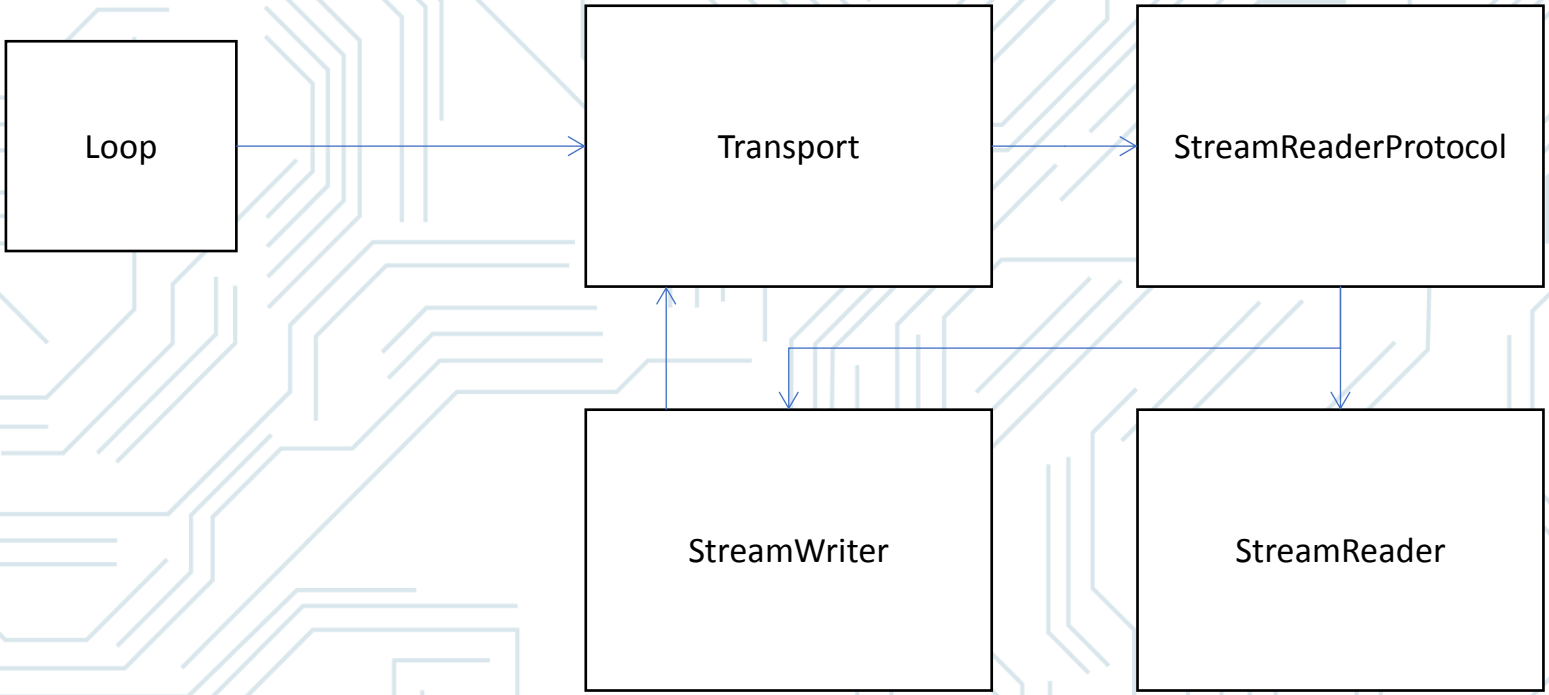
Piter Py #4

Asyncio StreamReaderProtocol



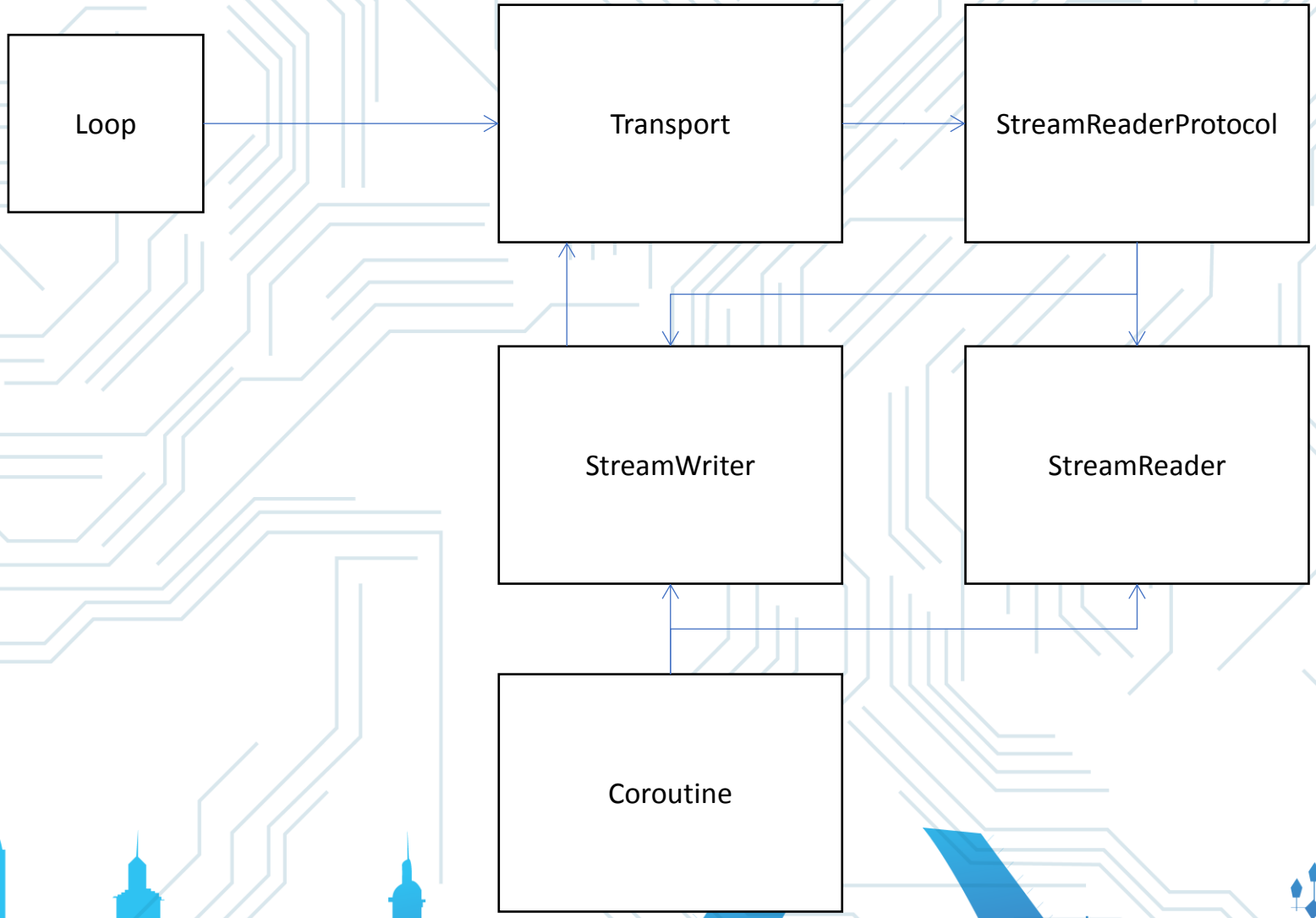
Piter Py #4

Asyncio StreamReaderProtocol



Piter Py #4

Asyncio StreamReaderProtocol



Enhancements

- More control over SMTP commands

Enhancements

- More control over SMTP commands
- Context between commands

Enhancements

- More control over SMTP commands
- Context between commands
- Encryption

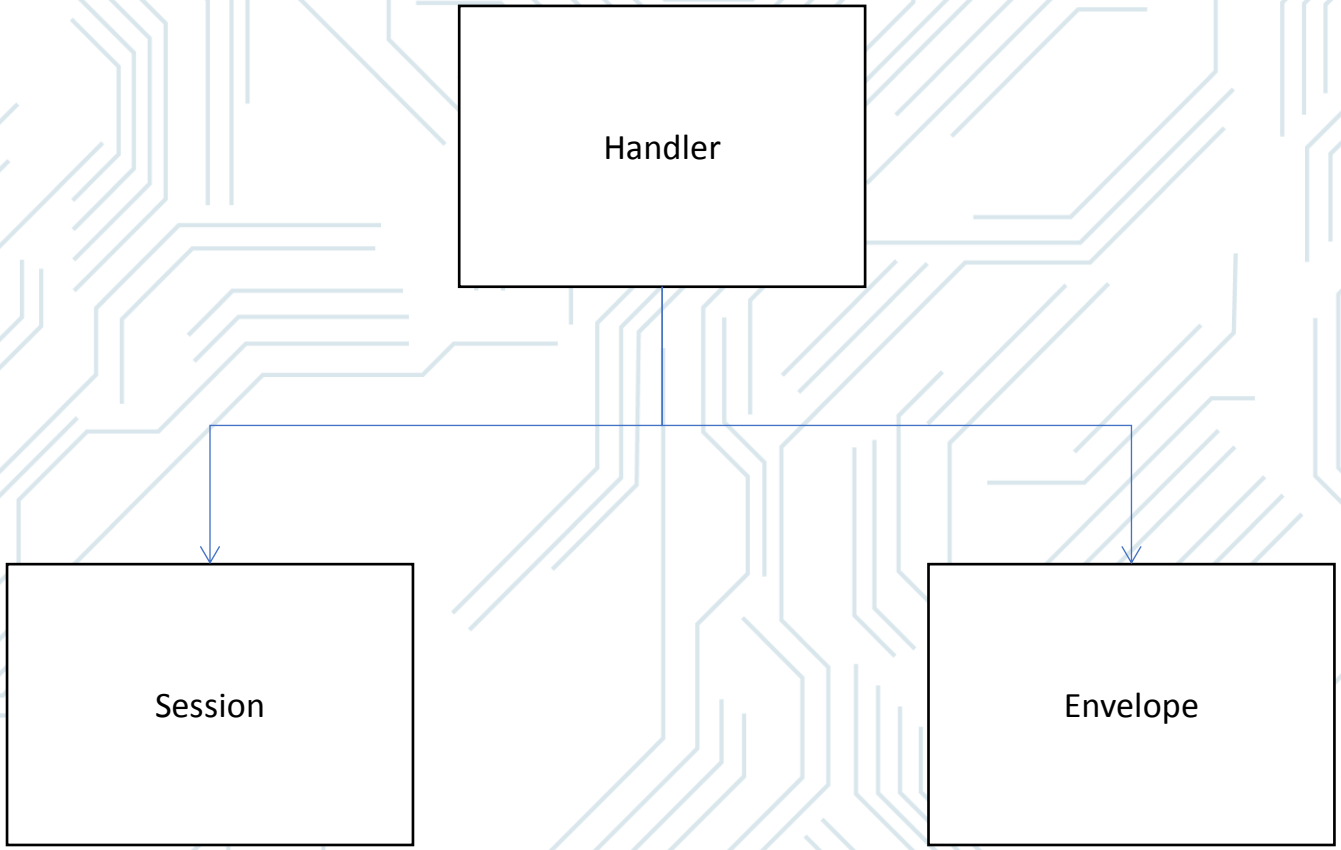
Enhancements

- More control over SMTP commands
- Context between commands
- Encryption
- Stability

Enhancements

- More control over SMTP commands
- Context between commands
- Encryption
- Stability
- Clear code

Additional entities

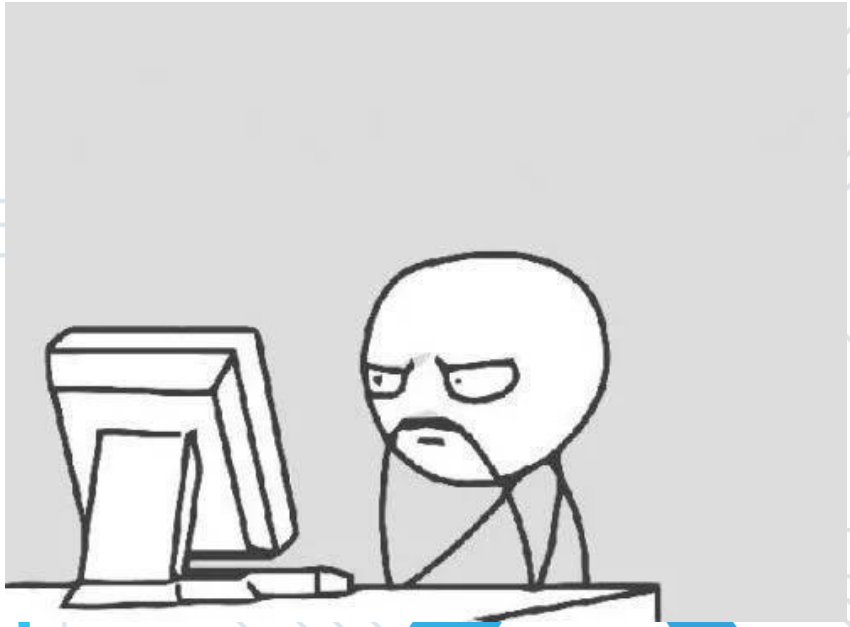


StartTLS

```
'''
STARTTLS implementation taken from Yury Selivanov
(http://bugs.python.org/issue23749).

This is a fragile implementation as it depends on asyncio internals that
may (will!) change. Hopefully, we can remove it in future.
'''
```

StartTLS

```
'''
STARTTLS implementation taken from Yury Selivanov
(http://bugs.python.org/issue23749).

This is a fragile implementation as it depends on asyncio internals that
may (will!) change. Hopefully, we can remove it in future.
'''
```

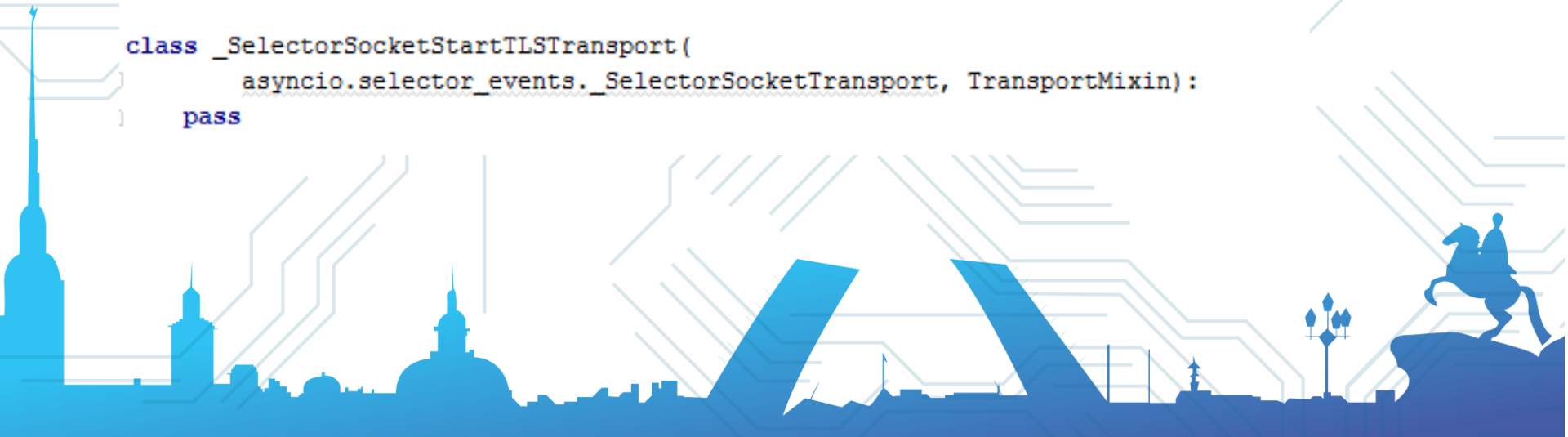


StartTLS

```
class SSLProtocol(asyncio.sslproto.SSLProtocol):

    def __init__(self, *args, call_connection_made=True, **kwargs):...

    def _on_handshake_complete(self, handshake_exc):
        self._in_handshake = False

        sslobj = self._sslpipe.ssl_object
        try:
            if handshake_exc is not None:
                raise handshake_exc

            peercert = sslobj.getpeercert()
        except BaseException as exc:...

        if self._loop.get_debug():
            dt = self._loop.time() - self._handshake_start_time
            logger.debug("%r: SSL handshake took %.1f ms", self, dt * 1e3)

        # Add extra info that becomes available after handshake.
        self._extra.update(peercert=peercert,
                           cipher=sslobj.cipher(),
                           compression=sslobj.compression(),
                           ssl_object=sslobj,
                           )
        if self._call_connection_made:
            self._app_protocol.connection_made(self._app_transport)
        self._wakeup_waiter()
        self._session_established = True
```

StartTLS

```
class TransportMixin(asyncio.Transport):

    def start_tls(self, sslcontext, *, server_side=False, server_hostname=None,
                  waiter=None):

        app_protocol = self._protocol
        ssl_protocol = SSLProtocol(
            loop=self._loop, app_protocol=app_protocol,
            sslcontext=sslcontext, waiter=waiter, server_side=server_side,
            server_hostname=server_hostname, call_connection_made=False)

        self._protocol = ssl_protocol
        ssl_protocol.connection_made(self)
        return ssl_protocol._app_transport


class _SelectorSocketStartTLSTransport(
        asyncio.selector_events._SelectorSocketTransport, TransportMixin):
    pass
```



StartTLS

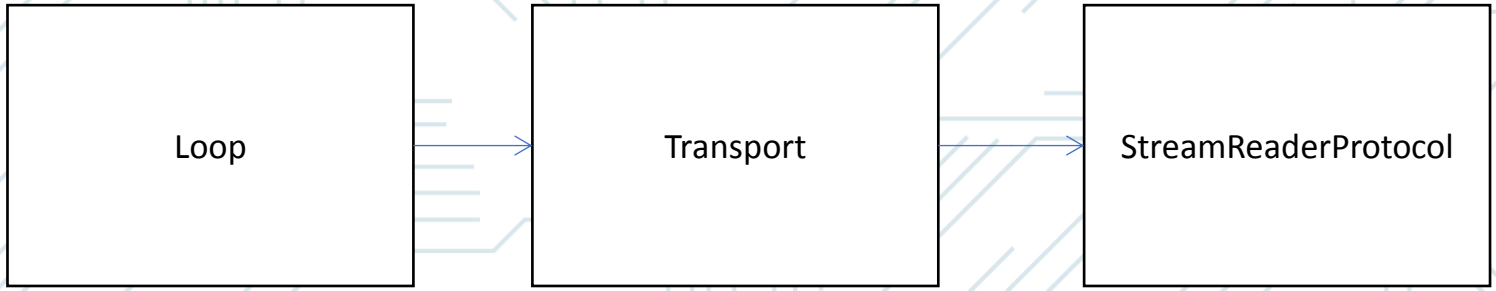
```
class StreamReaderProtocol(asyncio.StreamReaderProtocol):

    def start_tls(self, sslcontext, *, server_side=False,
                  server_hostname=None, waiter=None):

        transport = self._stream_reader._transport
        new_transport = transport.start_tls(
            sslcontext, server_side=server_side,
            server_hostname=server_hostname, waiter=waiter)

        self._stream_reader._transport = new_transport
        if self._stream_writer is not None:
            self._stream_writer._transport = new_transport

        return new_transport


class StreamWriter(asyncio.StreamWriter):

    async def start_tls(self, sslcontext, *, server_side=False,
                        server_hostname=None):

        if not asyncio.sslproto._is_sslproto_available():
            # Python 3.5 or greater is required
            raise NotImplementedError

        await self.drain()
```

StartTLS

```
class StartTLSSelectorEventLoop(asyncio.SelectorEventLoop):

    def _make_socket_transport(self, sock, protocol, waiter=None, *,
                               extra=None, server=None):
        return _SelectorSocketStartTLSTransport(
            self, sock, protocol, waiter, extra, server)


class StreamReaderProtocol(asyncio.StreamReaderProtocol):...


class StreamWriter(asyncio.StreamWriter):...


# Globally install our custom event loop
asyncio.set_event_loop(StartTLSSelectorEventLoop(selectors.SelectSelector()))
```

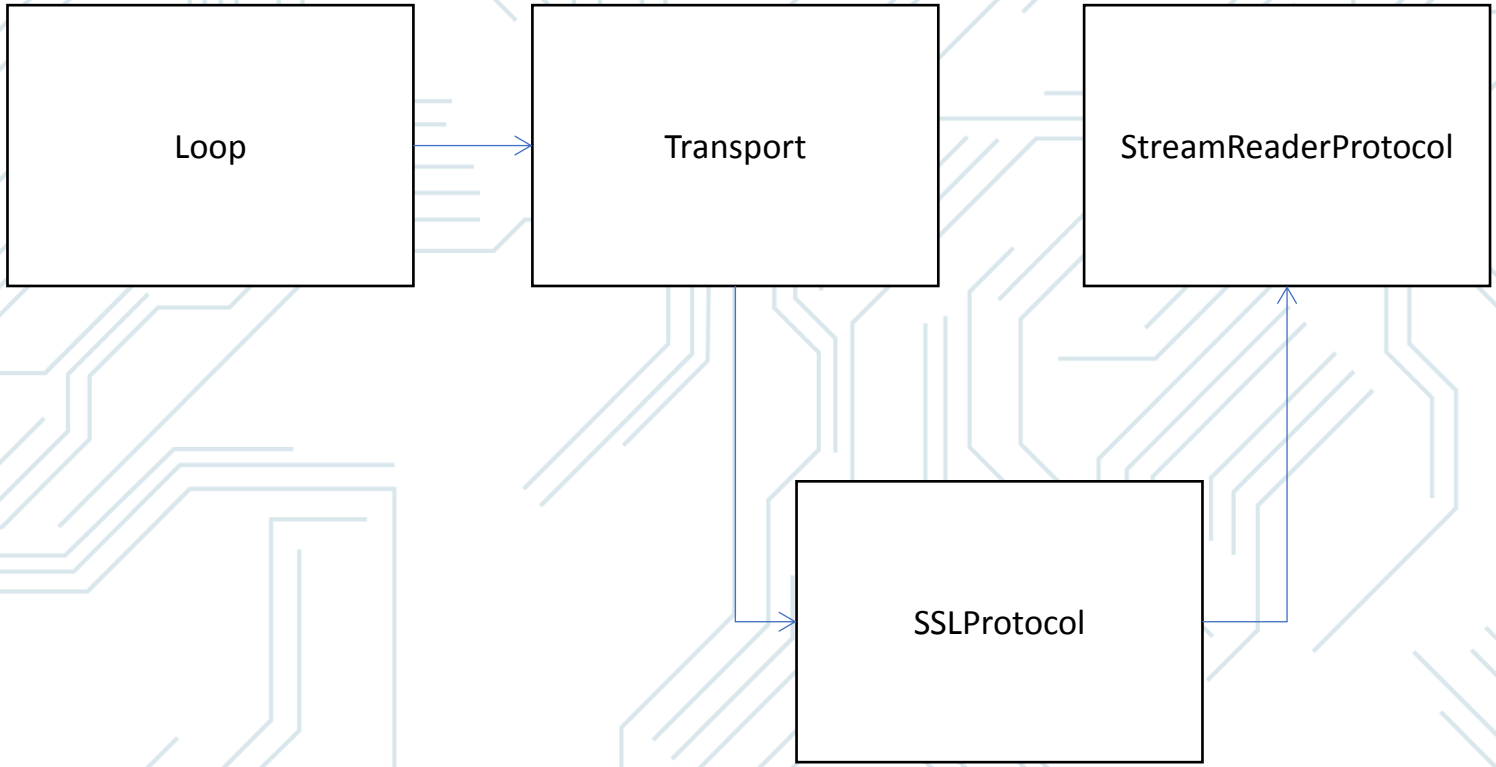
Piter Py #4

Asyncio encryption



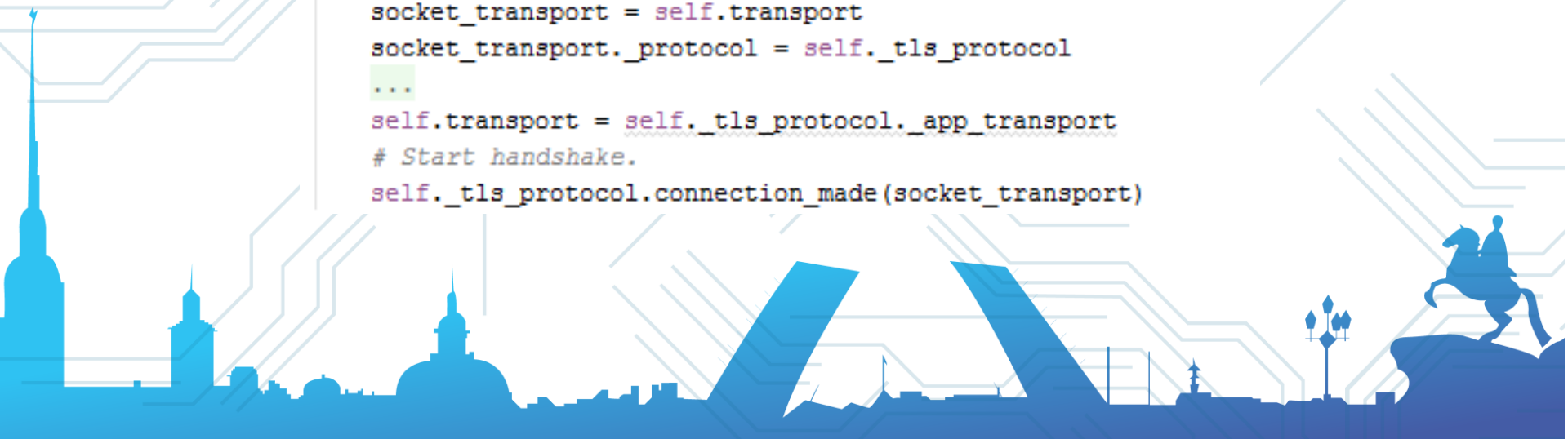
Piter Py #4

Asyncio encryption



StartTLS

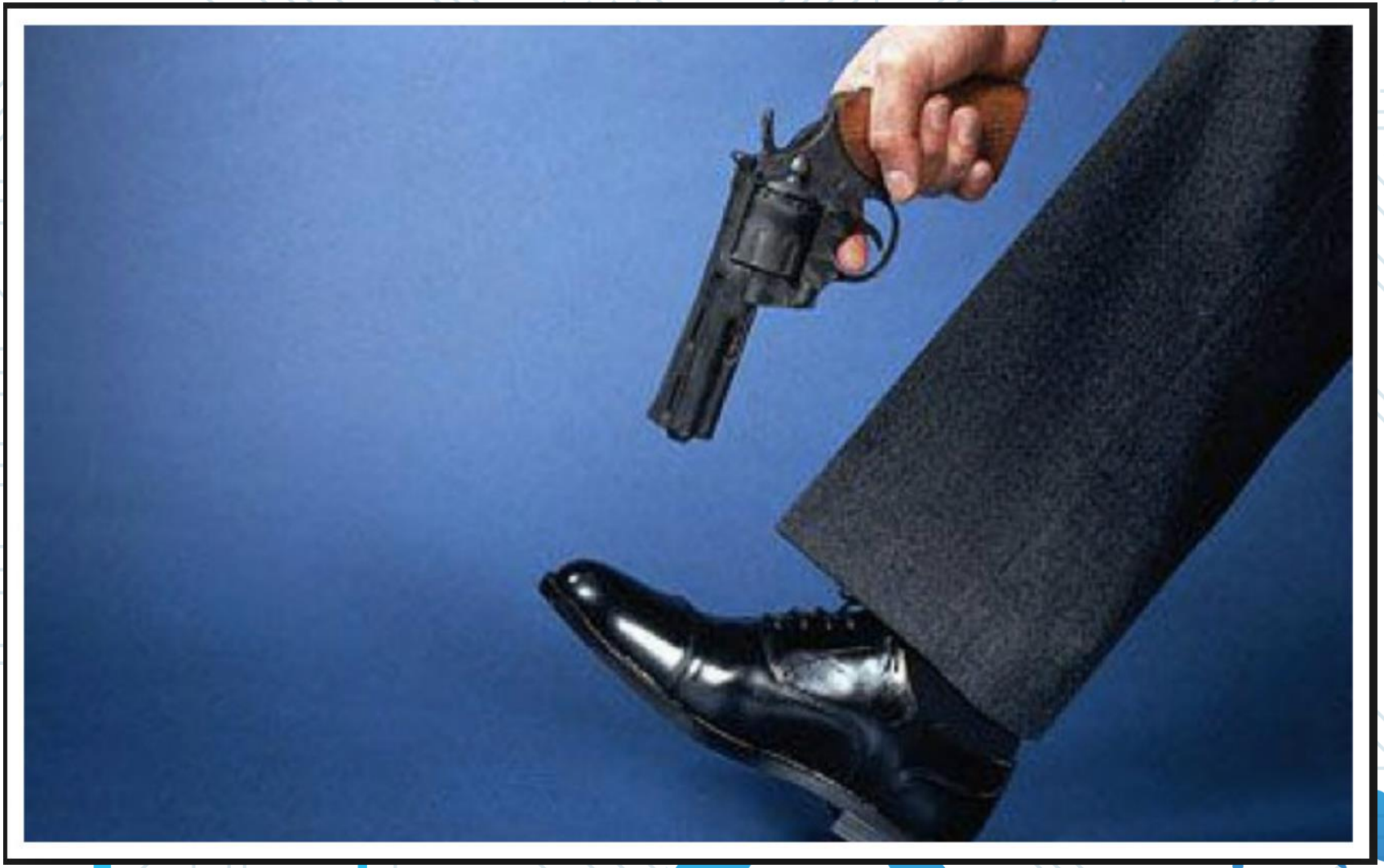
```
@asyncio.coroutine
def smtp_STARTTLS(self, arg):                    # pragma: nossl
    log.info('%r STARTTLS', self.session.peer)
    if arg:...
    if not (self.tls_context and _has_ssl):...
    yield from self.push('220 Ready to start TLS')
    # Create SSL layer.
    self._tls_protocol = sslproto.SSLProtocol(
        self.loop,
        self,
        self.tls_context,
        None,
        server_side=True)
    # Reconfigure transport layer.
    socket_transport = self.transport
    socket_transport._protocol = self._tls_protocol
    ...
    self.transport = self._tls_protocol._app_transport
    # Start handshake.
    self._tls_protocol.connection_made(socket_transport)
```



StartTLS

```
def connection_made(self, transport):
    # Reset state due to rfc3207 part 4.2.
    self._set_rset_state()
    self.session = self._create_session()
    self.session.peer = transport.get_extra_info('peername')
    is_instance = (_has_ssl and
                   isinstance(transport, sslproto._SSLProtocolTransport))
    if self.transport is not None and is_instance:    # pragma: nossl
        # It is STARTTLS connection over normal connection.
        self._reader._transport = transport
        self._writer._transport = transport
        self.transport = transport
        # Do SSL certificate checking as rfc3207 part 4.1 says.
        # Why _extra is protected attribute?
        self.session.ssl = self._tls_protocol._extra
        handler = getattr(self.event_handler, 'handle_STARTTLS', None)
        if handler is None:
            self._tls_handshake_okay = True
        else:
            self._tls_handshake_okay = handler(
                self, self.session, self.envelope)
    else:
        super().connection_made(transport)
        self.transport = transport
```

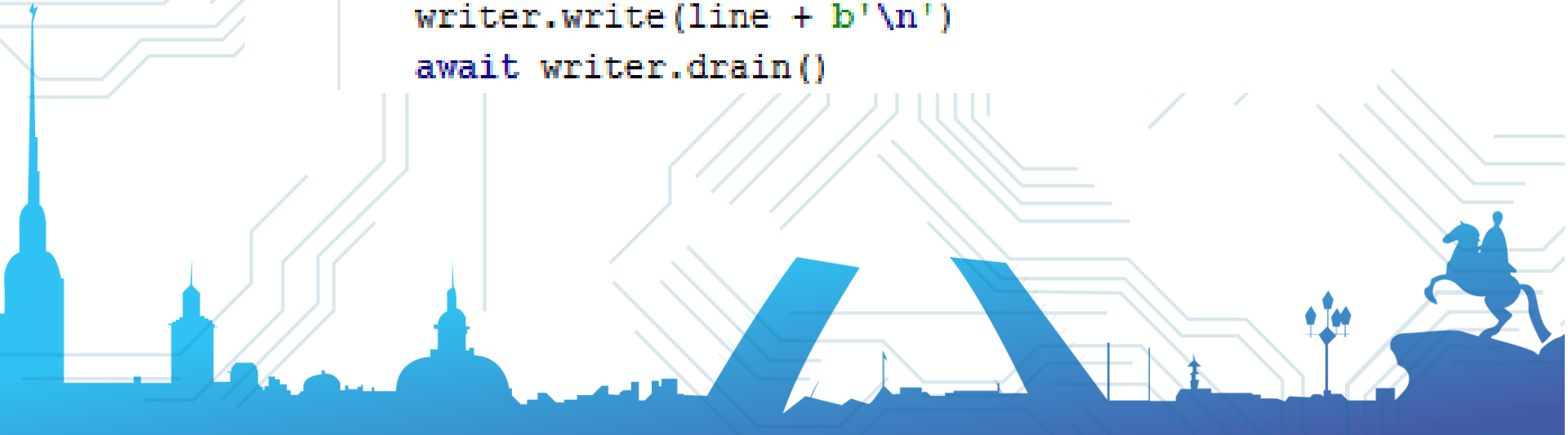
Power



Power

Closing connection

```
class MyServerProtocol(asyncio.StreamReaderProtocol):
    def __init__(self, loop=None):
        super(MyServerProtocol, self).__init__(
            asyncio.StreamReader(loop=loop),
            self.handler,
            loop=loop)

    @staticmethod
    async def handler(reader, writer):
        line = await reader.readline()
        writer.write(line + b'\n')
        await writer.drain()
```



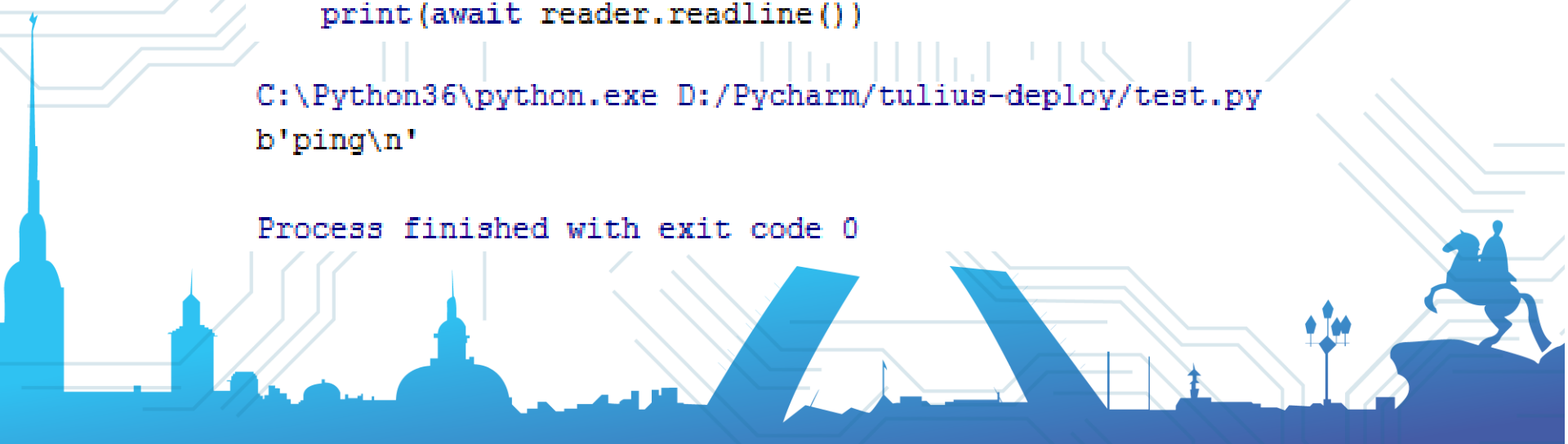
Closing connection

```
async def main(loop):
    server = await loop.create_server(MyServerProtocol, host='127.0.0.1', port=8000)
    try:
        reader, writer = await asyncio.open_connection('127.0.0.1', 8000, loop=loop)
        await client(reader, writer)
    finally:
        server.close()

event_loop = asyncio.get_event_loop()
event_loop.run_until_complete(main(event_loop))
```

Closing connection

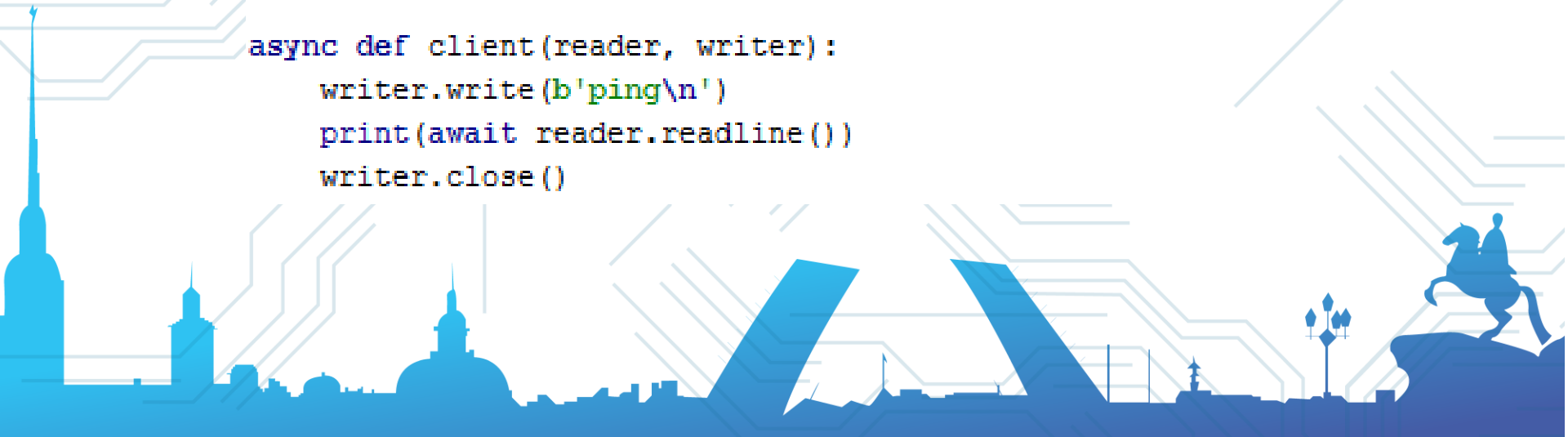
```
class MyServerProtocol(asyncio.StreamReaderProtocol):
    def __init__(self, loop=None):...

    @staticmethod
    async def handler(reader, writer):
        line = await reader.readline()
        writer.write(line + b'\n')
        await writer.drain()


async def client(reader, writer):
    writer.write(b'ping\n')
    print(await reader.readline())
```

```
C:\Python36\python.exe D:/Pycharm/tulius-deploy/test.py
b'ping\n'

Process finished with exit code 0
```



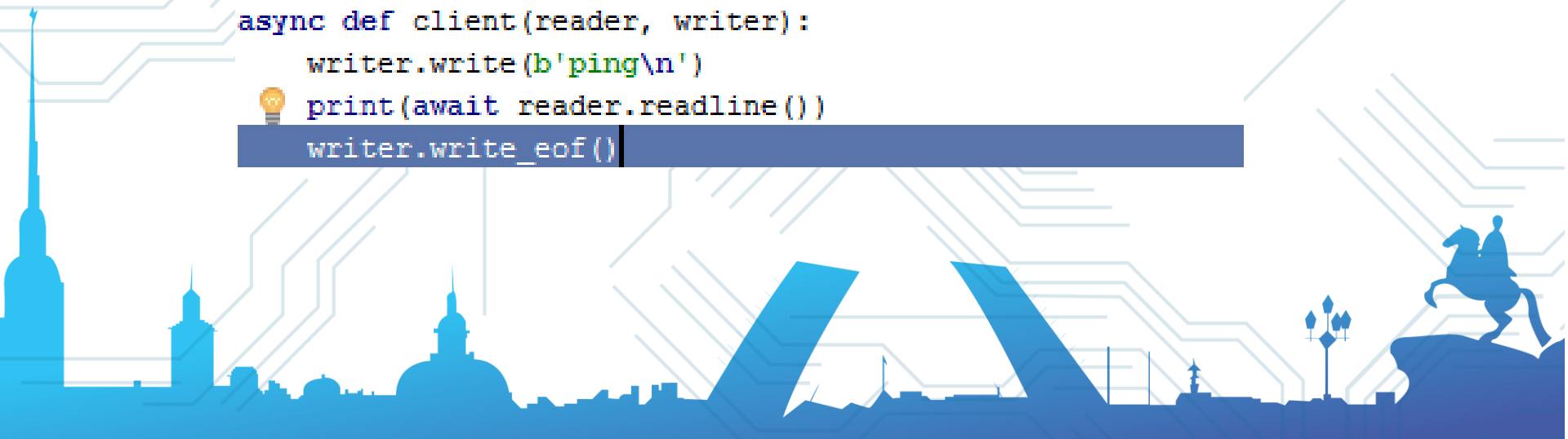
Closing connection

```
class MyServerProtocol(asyncio.StreamReaderProtocol):
    def __init__(self, loop=None):...

    @staticmethod
    async def handler(reader, writer):
        try:
            line = await reader.readline()
            writer.write(line + b'\n')
            await writer.drain()
        finally:
            writer.close()


async def client(reader, writer):
    writer.write(b'ping\n')
    print(await reader.readline())
    writer.close()
```



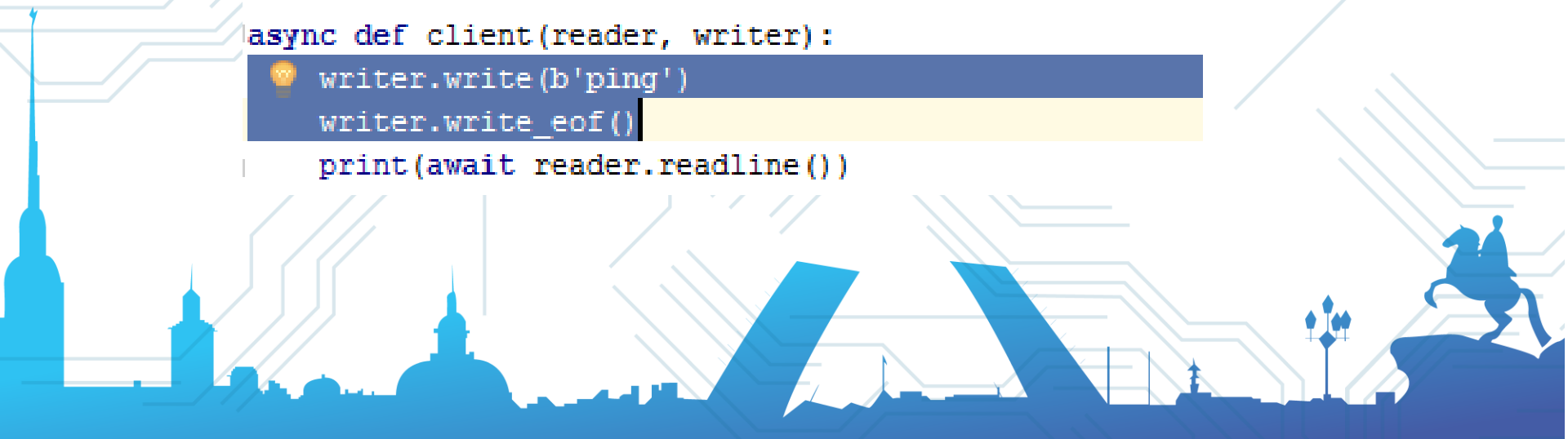
Closing connection

```
class MyServerProtocol(asyncio.StreamReaderProtocol):
    def __init__(self, loop=None):...

    @staticmethod
    async def handler(reader, writer):
        try:
            line = await reader.readline()
            writer.write(line + b'\n')
            await writer.drain()
        finally:
            writer.close()


async def client(reader, writer):
    writer.write(b'ping\n')
    print(await reader.readline())
    writer.write_eof()
```



Closing connection

```
class MyServerProtocol(asyncio.StreamReaderProtocol):
    def __init__(self, loop=None):...

    @staticmethod
    async def handler(reader, writer):
        try:
            line = await reader.readline()
            writer.write(line + b'\n')
            await writer.drain()
        finally:
            writer.close()


async def client(reader, writer):
    writer.write(b'ping')
    writer.write_eof()
    print(await reader.readline())
```



Closing connection

```
class MyServerProtocol(asyncio.StreamReaderProtocol):
    def __init__(self, loop=None):...

    @staticmethod
    async def handler(reader, writer):
        try:
            line = await reader.readuntil(b'\n')
            writer.write(line + b'\n')
            await writer.drain()
        finally:
            writer.close()


async def client(reader, writer):
    writer.write(b'ping')
    writer.write_eof()
    print(await reader.readline())
```

Patching asyncio

```
import asyncio


class MyTask(asyncio):
    def set_exception(self, exception):
        super(MyTask, self).set_exception(exception)
        print('i am here')

    @classmethod
    def factory(cls, loop, coro):
        return cls(coro, loop=loop)
```

Patching asyncio

```
import asyncio


class MyTask(asyncio):
    def set_exception(self, exception):
        super(MyTask, self).set_exception(exception)
        print('i am here')

    @classmethod
    def factory(cls, loop, coro):
        return cls(coro, loop=loop)

loop = asyncio.get_event_loop()
loop.set_task_factory(MyTask.factory)


async def func():
    raise Exception('Big badaboom')

try:
   loop.run_until_complete(func())
except:
    pass
```

Patching asyncio

```
_PyTask = Task


try:
    import _asyncio
except ImportError:
    pass
else:
    # _CTask is needed for tests.
    Task = _CTask = _asyncio.Task
```

Patching asyncio

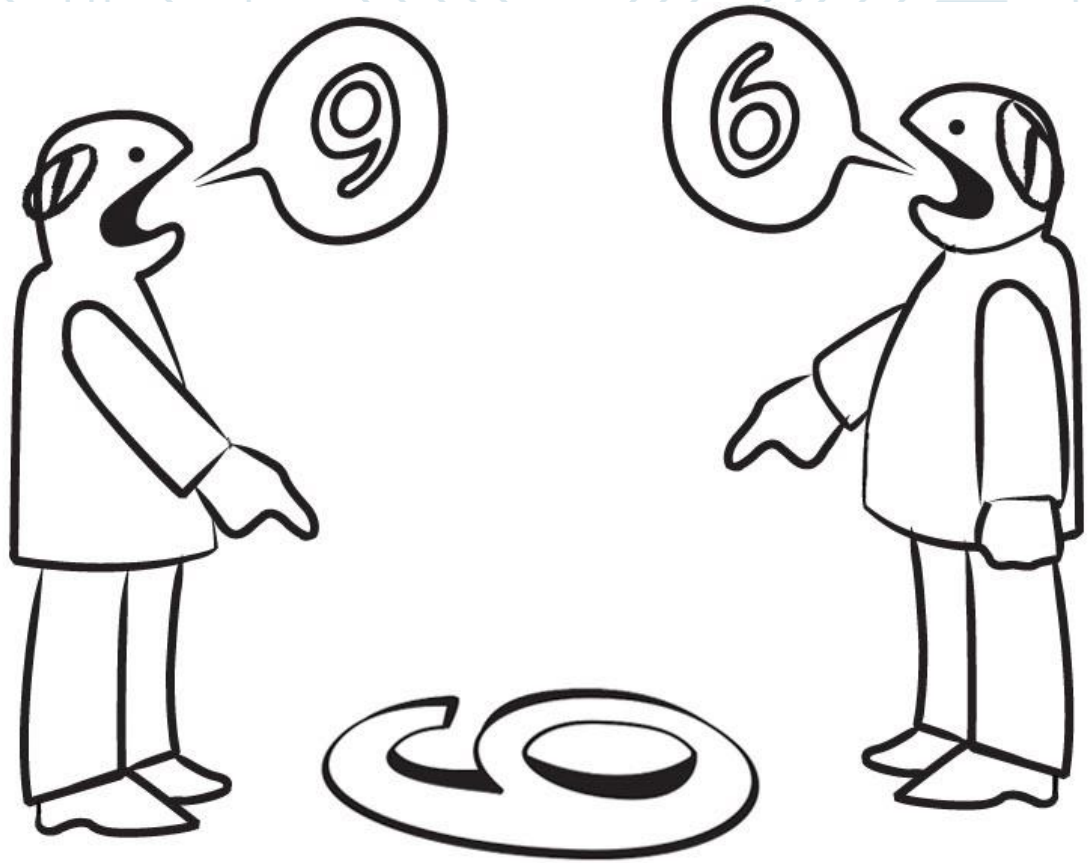
```
import asyncio.tasks


class MyTask(asyncio.tasks._PyTask):
    def set_exception(self, exception):
        super(MyTask, self).set_exception(exception)
        print('i am here')

    @classmethod
    def factory(cls, loop, coro):
        return cls(coro, loop=loop)
```

Disconnects

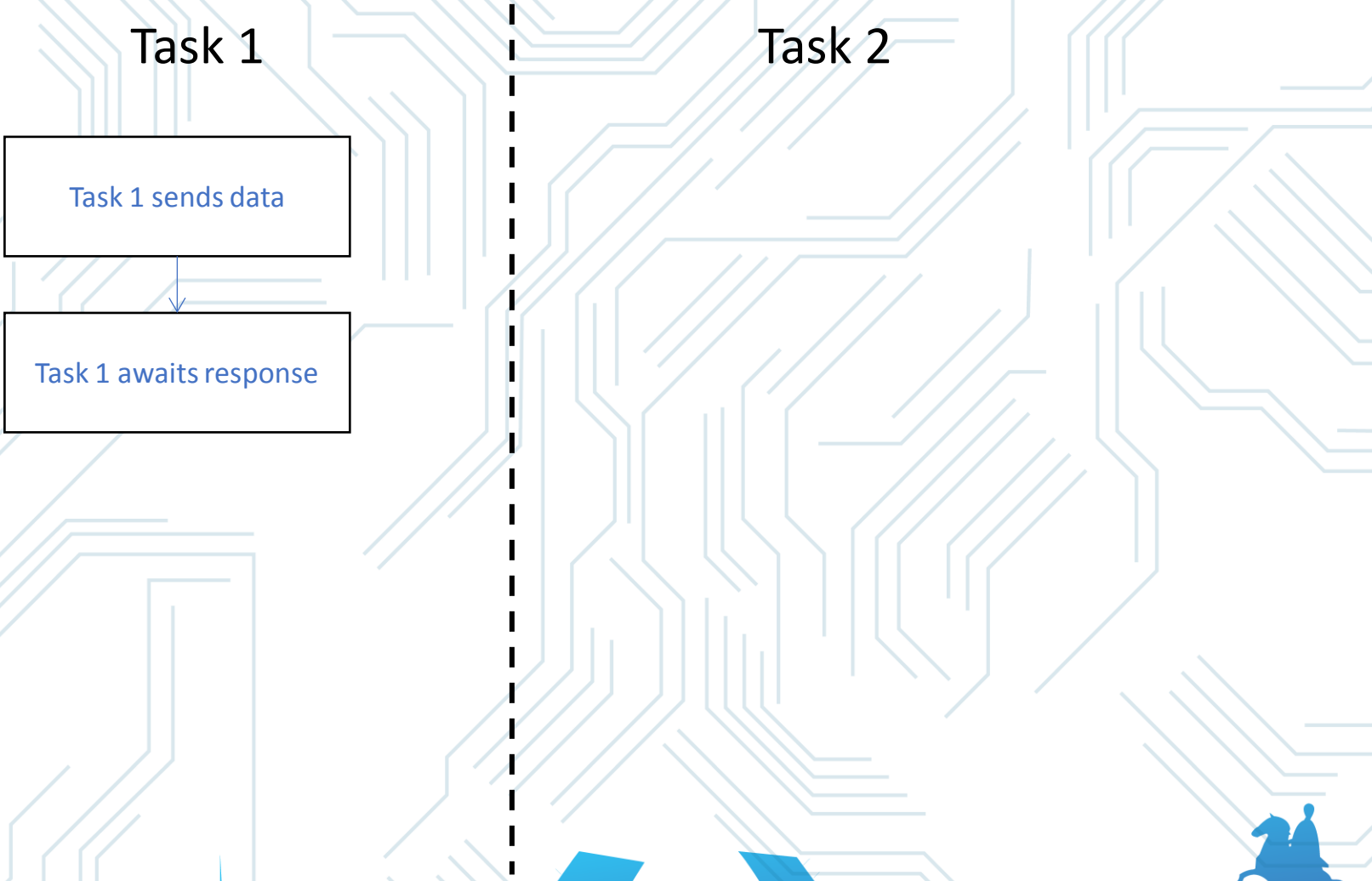
- Request timeouts for external service
- Happens on same period every day
- Of cource Friday evening affected
- No problems in monitoring

Piter Py #4

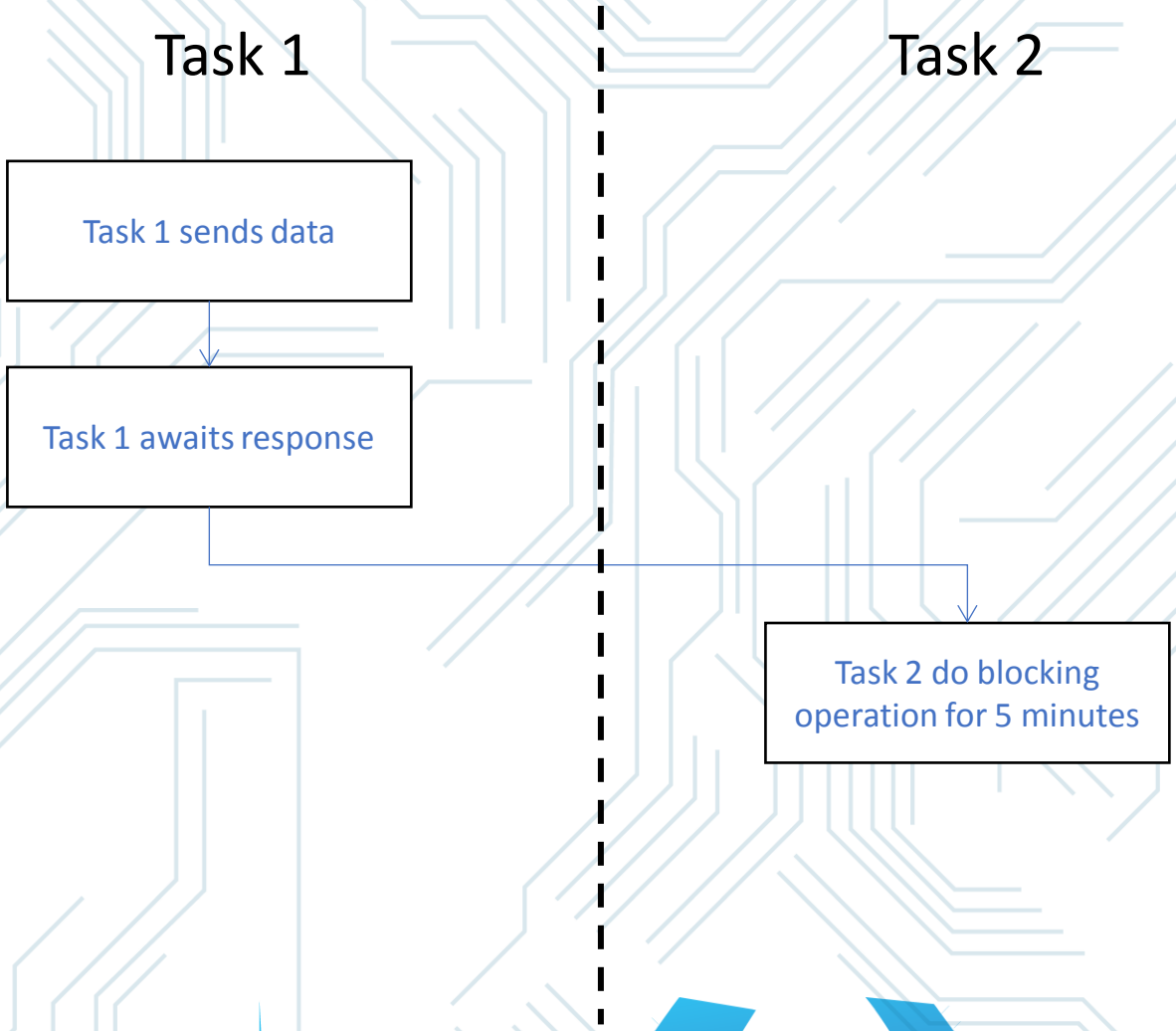
Disconnects



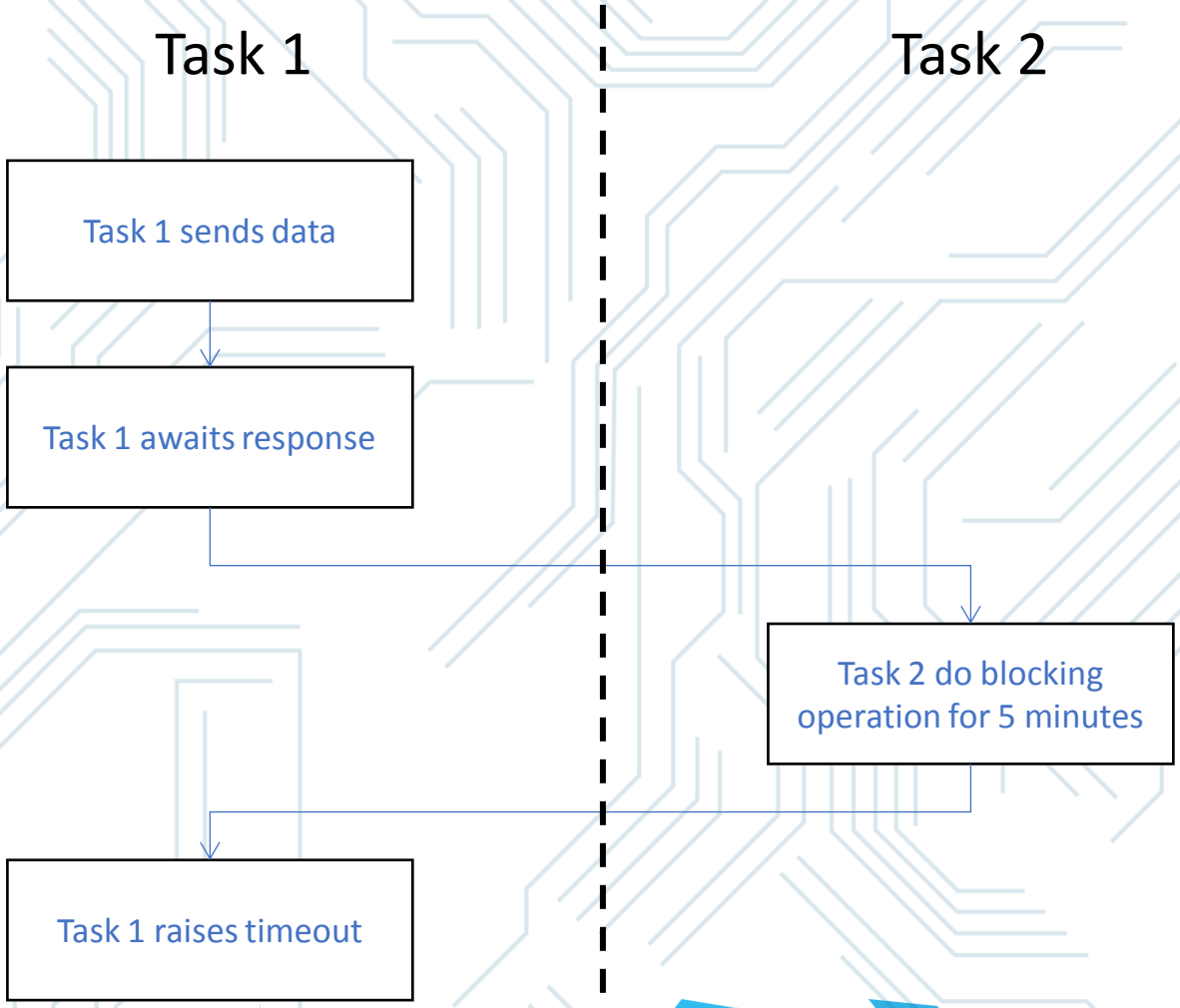
Loop concurrency



Loop concurrency



Loop concurrency



Piter Py #4

Pull requests

- 21 pull request
- 180+ comments

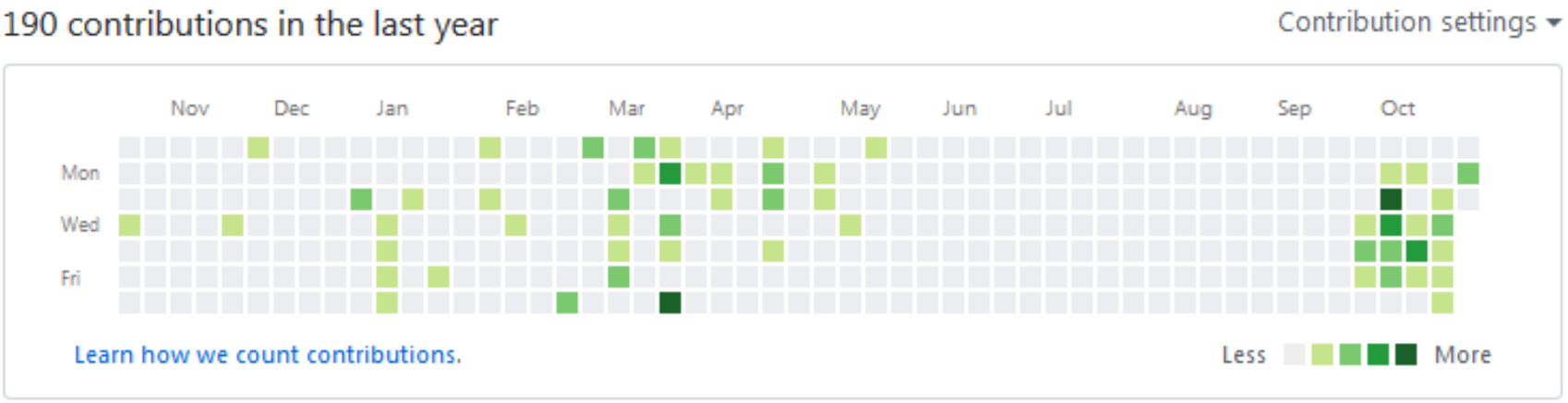
Piter Py #4

Pull requests

- 21 pull request
- 180+ comments
- 10 PRs was merged
- 5 PRs substituted

Piter Py #4

Contribute



QUIT

Questions?