MYE017 Distributed Systems

Kostas Magoutis

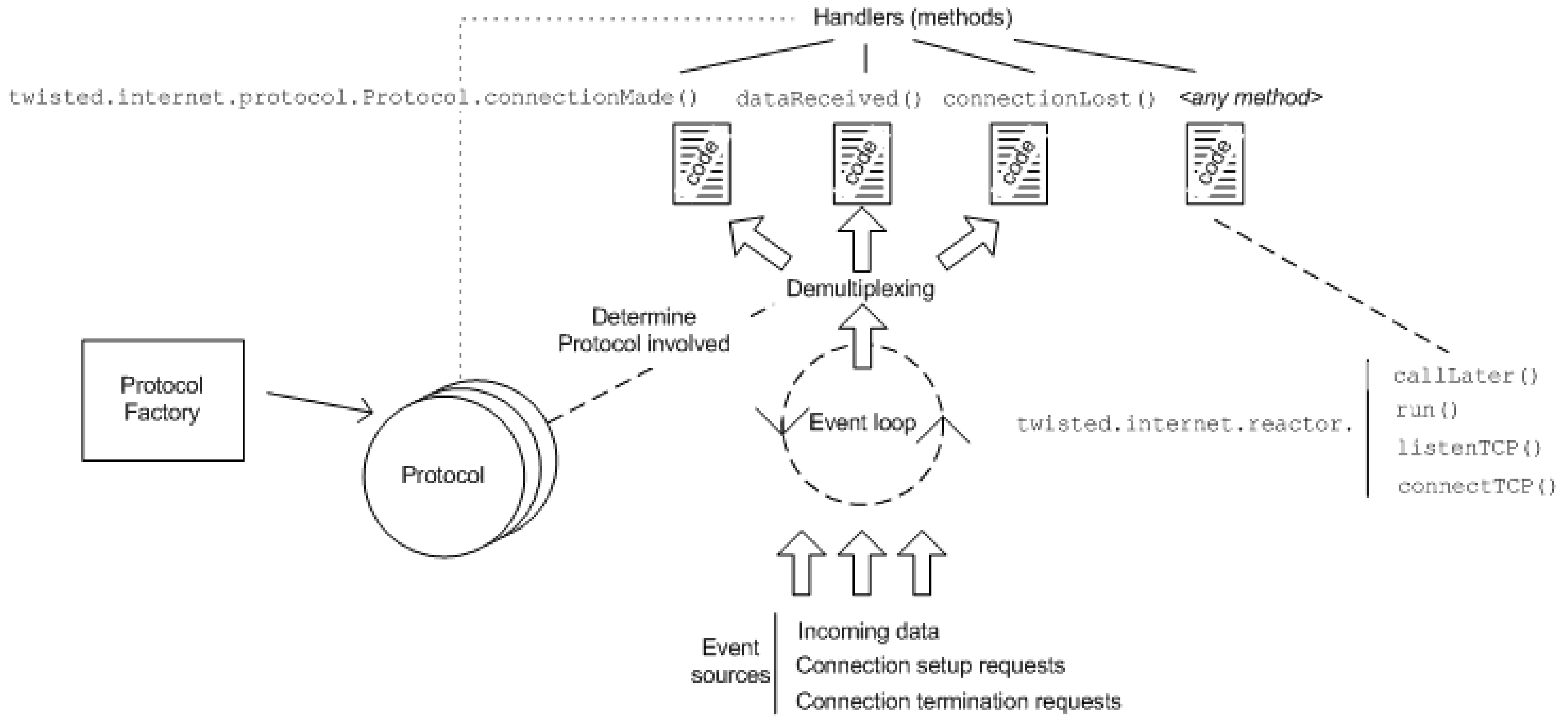
magoutis@cse.uoi.gr

http://www.cse.uoi.gr/~magoutis

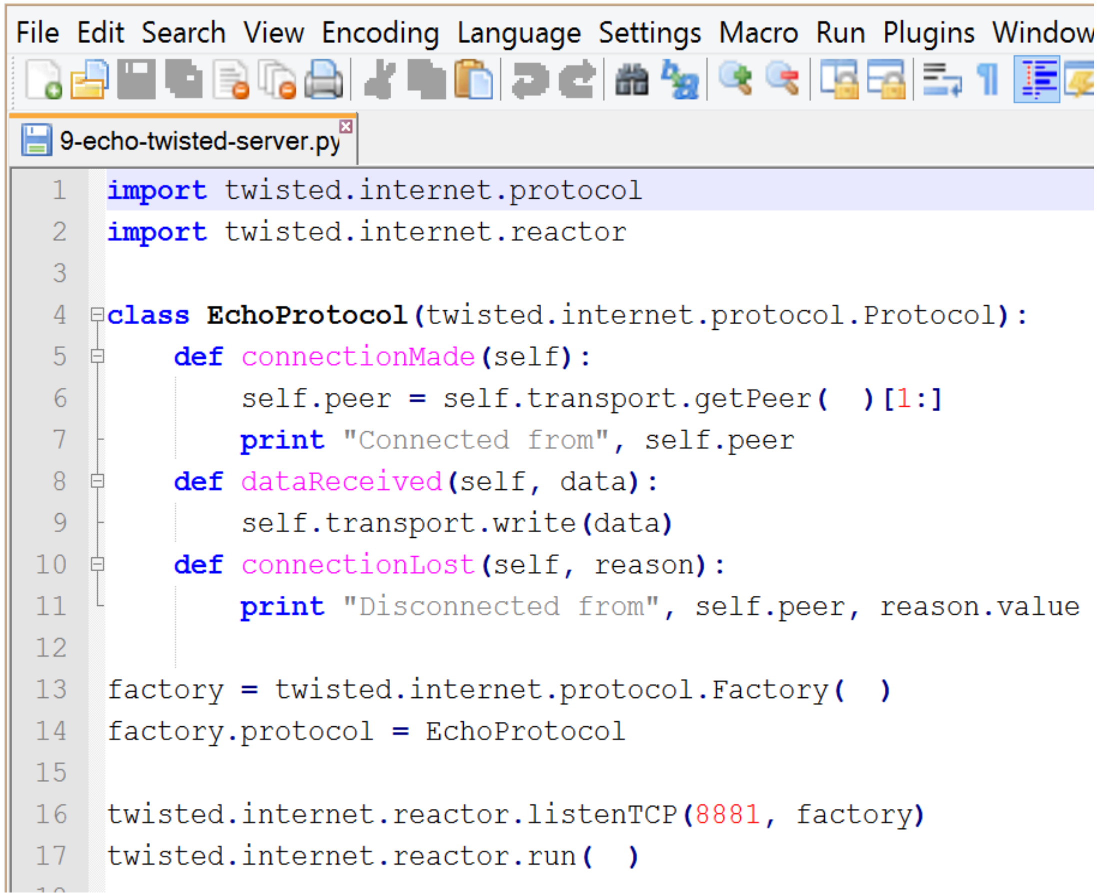
Twisted

- Event-driven communication framework
- Python module
- Rapid prototyping of distributed applications

Twisted event loop

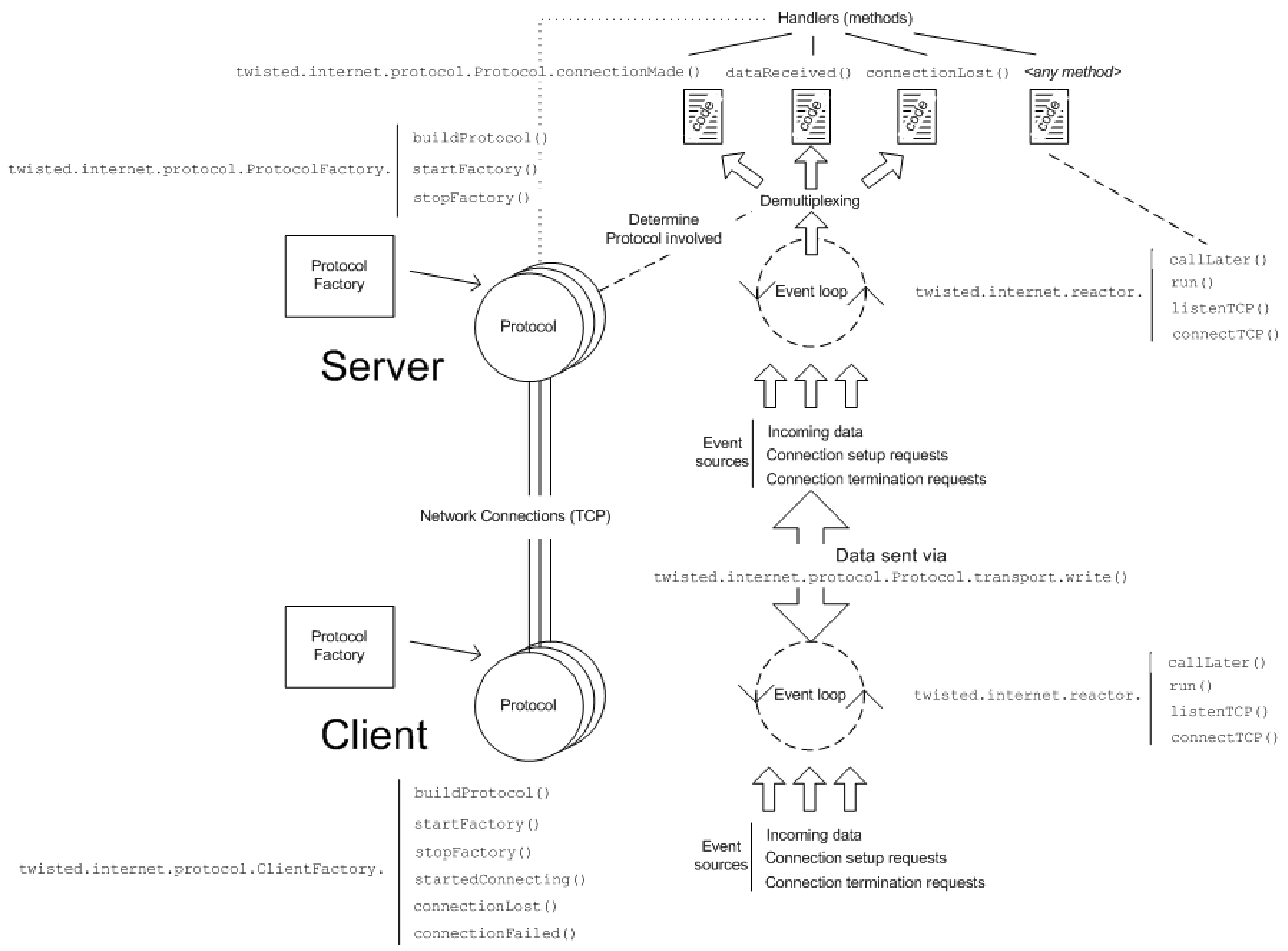


Echo server

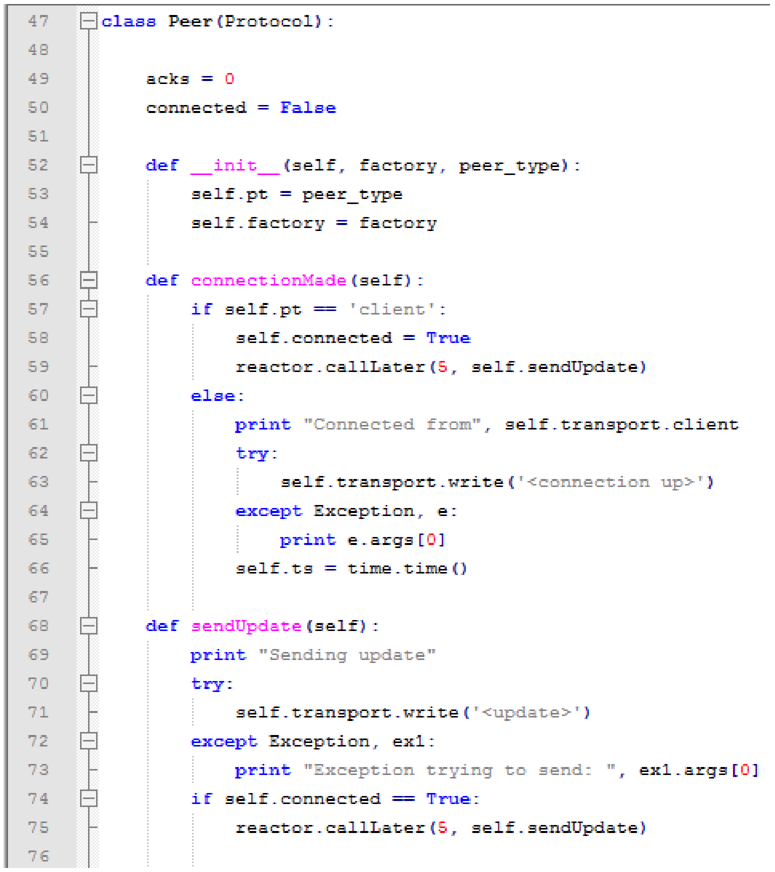

File Edit Search View Encoding Language Settings Macro Run Plugins Window
9-echo-twisted-server.py

```python
import twisted.internet.protocol
import twisted.internet.reactor

class EchoProtocol(twisted.internet.protocol.Protocol):
    def connectionMade(self):
        self.peer = self.transport.getPeer(  )[1:]
        print "Connected from", self.peer
    def dataReceived(self, data):
        self.transport.write(data)
    def connectionLost(self, reason):
        print "Disconnected from", self.peer, reason.value

factory = twisted.internet.protocol.Factory(  )
factory.protocol = EchoProtocol

twisted.internet.reactor.listenTCP(8881, factory)
twisted.internet.reactor.run(  )
```

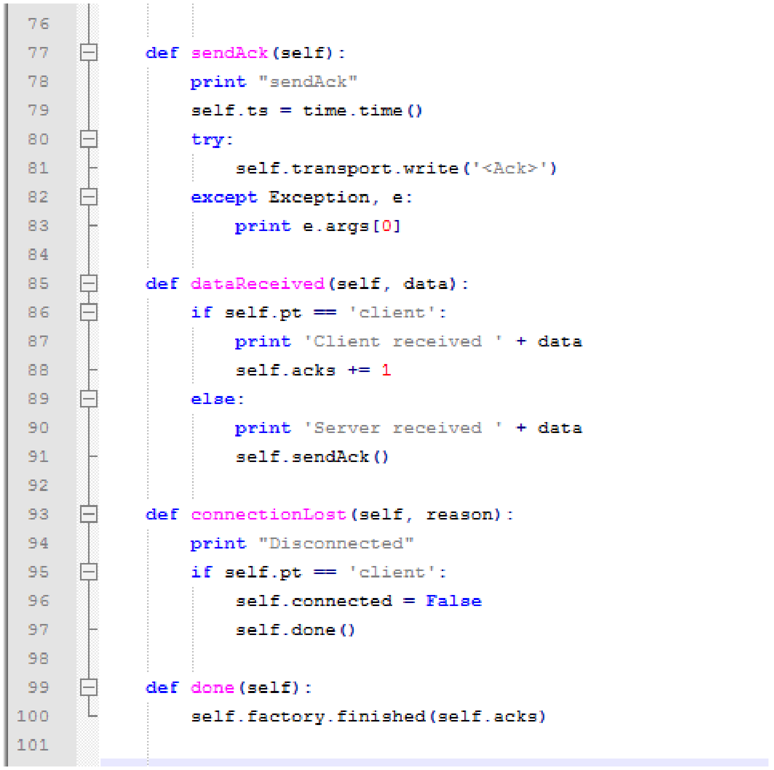
Twisted event loop (2)



Peer protocol

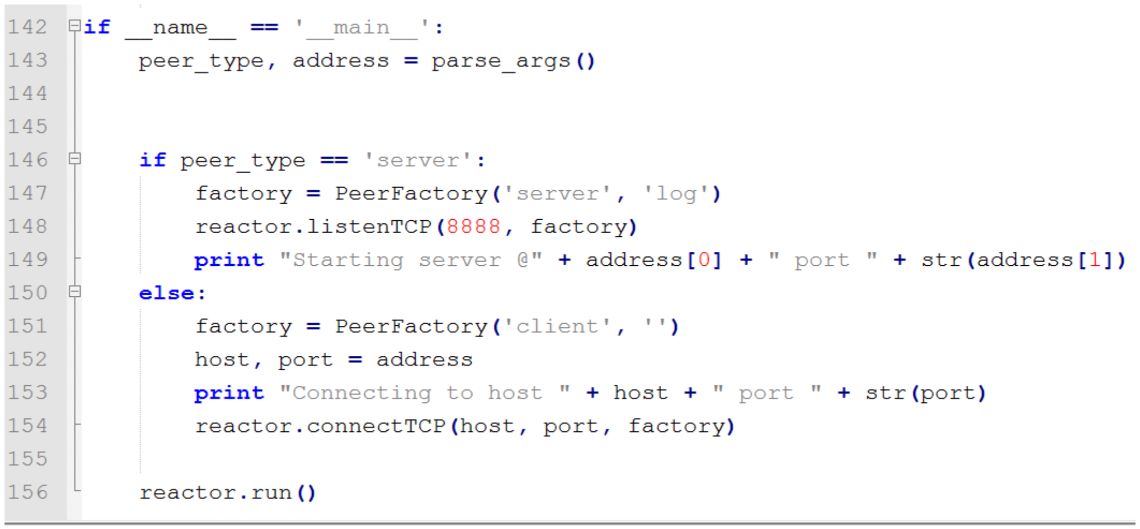


```python
class Peer(Protocol):

    acks = 0
    connected = False

    def __init__(self, factory, peer_type):
        self.pt = peer_type
        self.factory = factory

    def connectionMade(self):
        if self.pt == 'client':
            self.connected = True
            reactor.callLater(5, self.sendUpdate)
        else:
            print "Connected from", self.transport.client
            try:
                self.transport.write('<connection up>')
            except Exception, e:
                print e.args[0]
            self.ts = time.time()

    def sendUpdate(self):
        print "Sending update"
        try:
            self.transport.write('<update>')
        except Exception, ex1:
            print "Exception trying to send: ", ex1.args[0]
        if self.connected == True:
            reactor.callLater(5, self.sendUpdate)

```



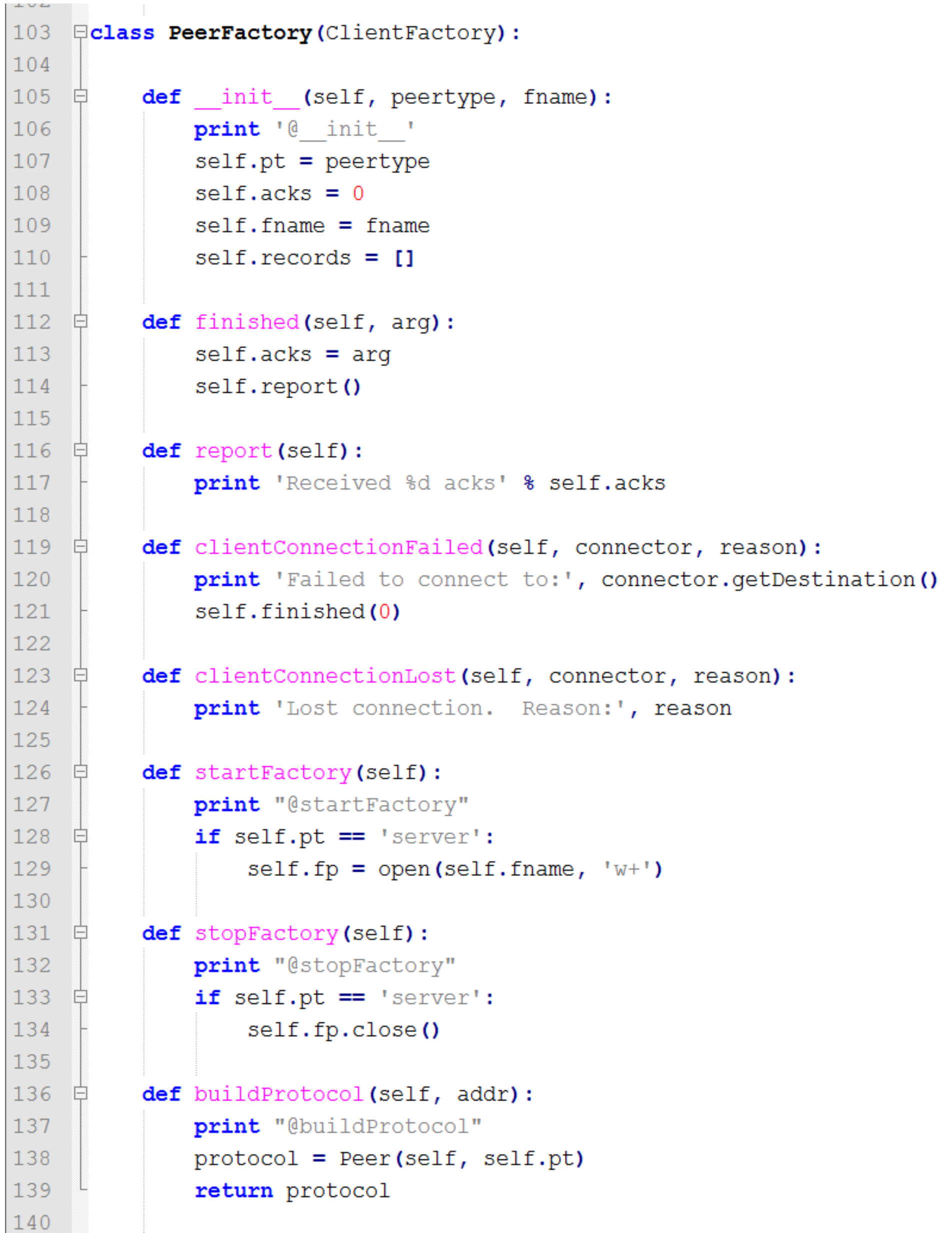
```python

    def sendAck(self):
        print "sendAck"
        self.ts = time.time()
        try:
            self.transport.write('<Ack>')
        except Exception, e:
            print e.args[0]

    def dataReceived(self, data):
        if self.pt == 'client':
            print 'Client received ' + data
            self.acks += 1
        else:
            print 'Server received ' + data
            self.sendAck()

    def connectionLost(self, reason):
        print "Disconnected"
        if self.pt == 'client':
            self.connected = False
            self.done()

    def done(self):
        self.factory.finished(self.acks)

```

Main program



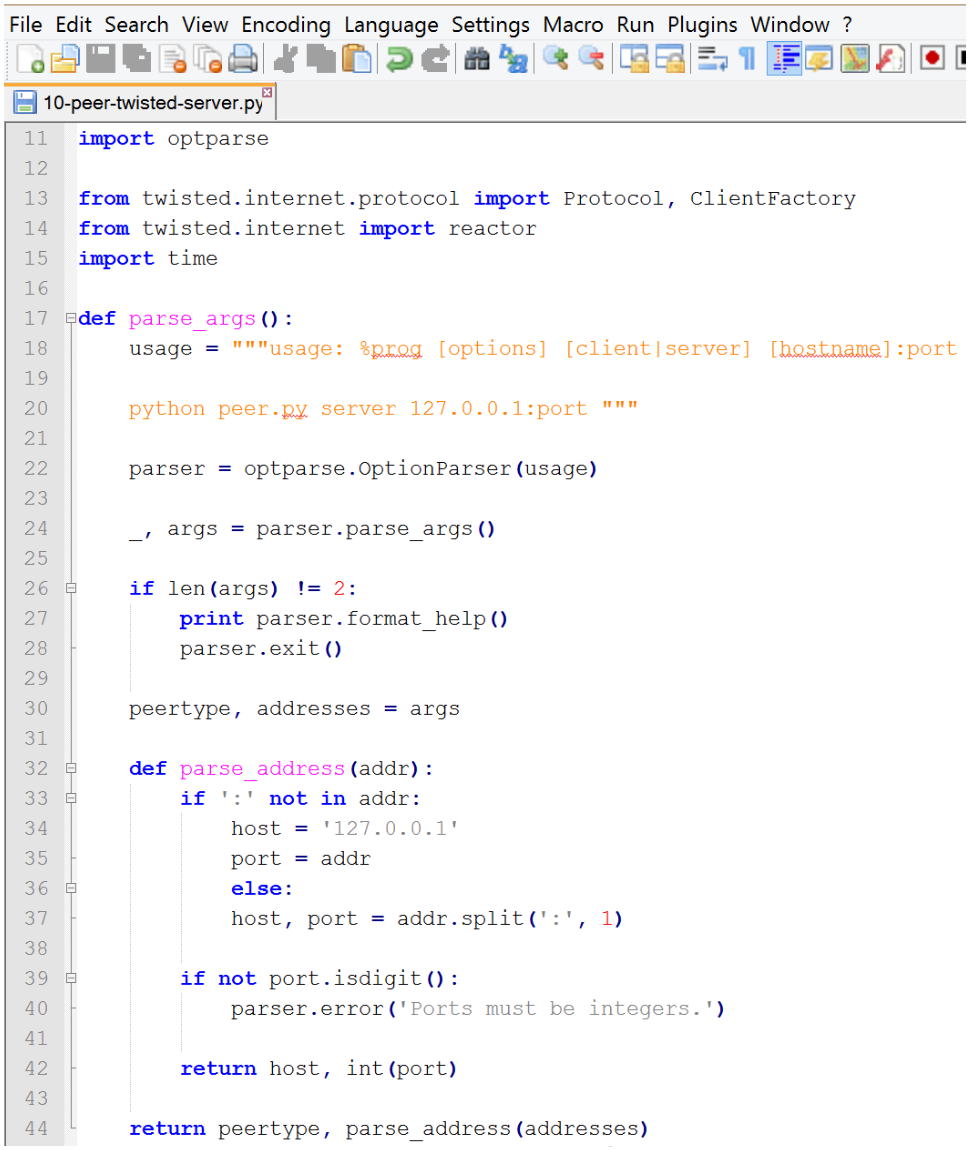
```python
if __name__ == '__main__':
    peer_type, address = parse_args()


    if peer_type == 'server':
        factory = PeerFactory('server', 'log')
        reactor.listenTCP(8888, factory)
        print "Starting server @" + address[0] + " port " + str(address[1])
    else:
        factory = PeerFactory('client', '')
        host, port = address
        print "Connecting to host " + host + " port " + str(port)
        reactor.connectTCP(host, port, factory)

    reactor.run()
```

Peer factory



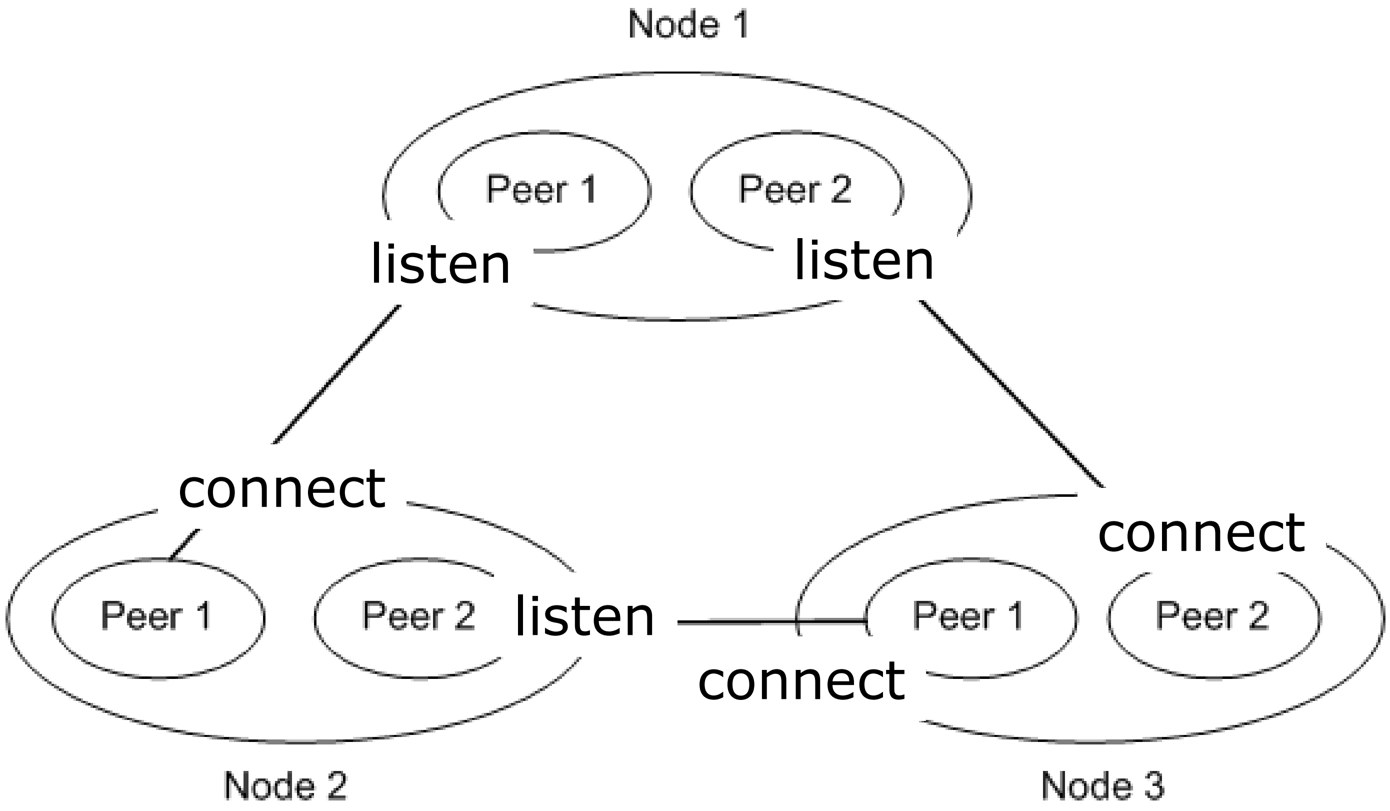
```python
class PeerFactory(ClientFactory):

    def __init__(self, peertype, fname):
        print '@__init__'
        self.pt = peertype
        self.acks = 0
        self.fname = fname
        self.records = []

    def finished(self, arg):
        self.acks = arg
        self.report()

    def report(self):
        print 'Received %d acks' % self.acks

    def clientConnectionFailed(self, connector, reason):
        print 'Failed to connect to:', connector.getDestination()
        self.finished(0)

    def clientConnectionLost(self, connector, reason):
        print 'Lost connection.  Reason:', reason

    def startFactory(self):
        print "@startFactory"
        if self.pt == 'server':
            self.fp = open(self.fname, 'w+')

    def stopFactory(self):
        print "@stopFactory"
        if self.pt == 'server':
            self.fp.close()

    def buildProtocol(self, addr):
        print "@buildProtocol"
        protocol = Peer(self, self.pt)
        return protocol

```

Program options

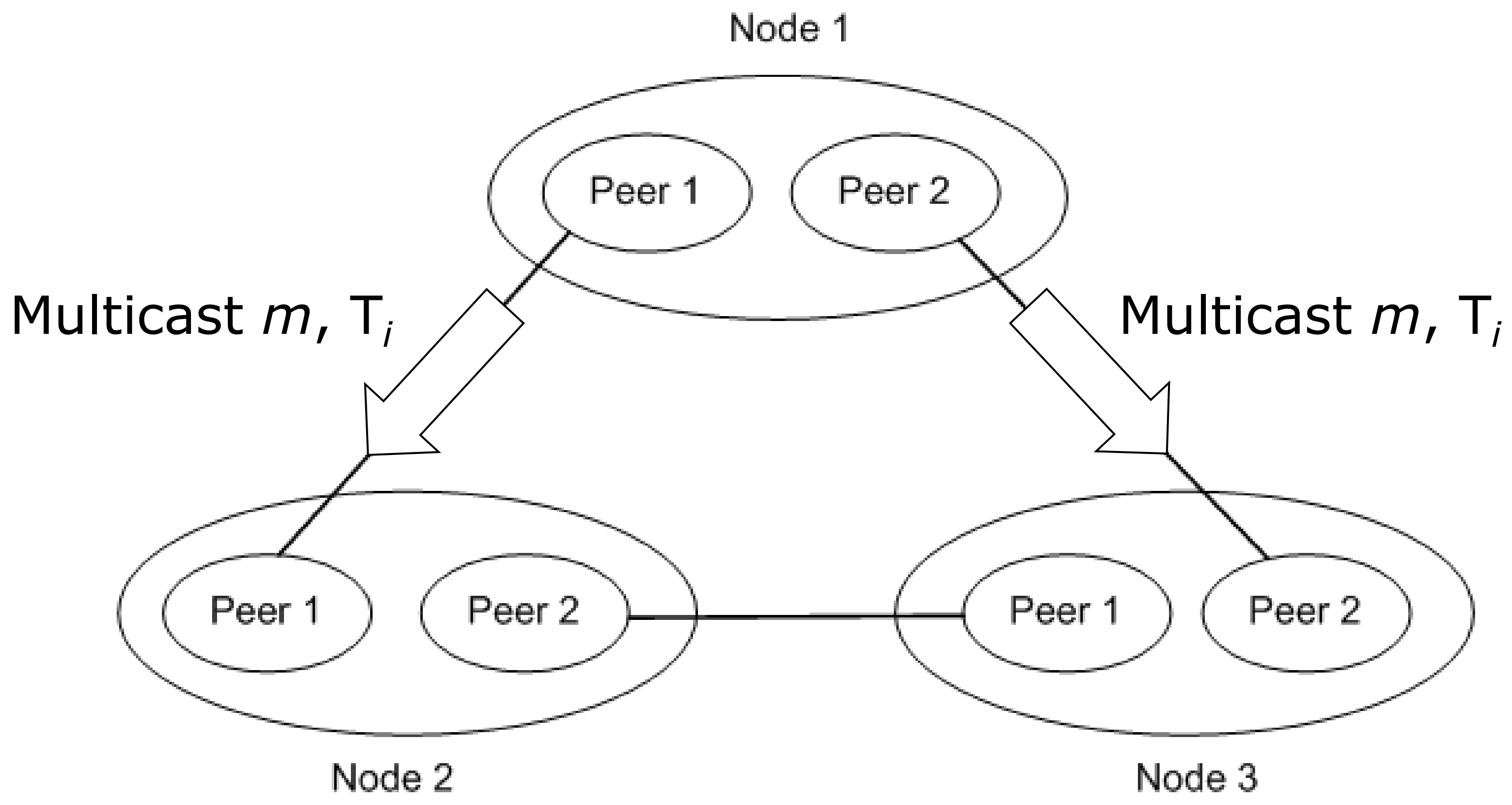

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
10-peer-twisted-server.py

```python
import optparse

from twisted.internet.protocol import Protocol, ClientFactory
from twisted.internet import reactor
import time

def parse_args():
    usage = """usage: %prog [options] [client|server] [hostname]:port

    python peer.py server 127.0.0.1:port """

    parser = optparse.OptionParser(usage)

    _, args = parser.parse_args()

    if len(args) != 2:
        print parser.format_help()
        parser.exit()

    peertype, addresses = args

    def parse_address(addr):
        if ':' not in addr:
            host = '127.0.0.1'
            port = addr
        else:
            host, port = addr.split(':', 1)

        if not port.isdigit():
            parser.error('Ports must be integers.')

        return host, int(port)

    return peertype, parse_address(addresses)
```

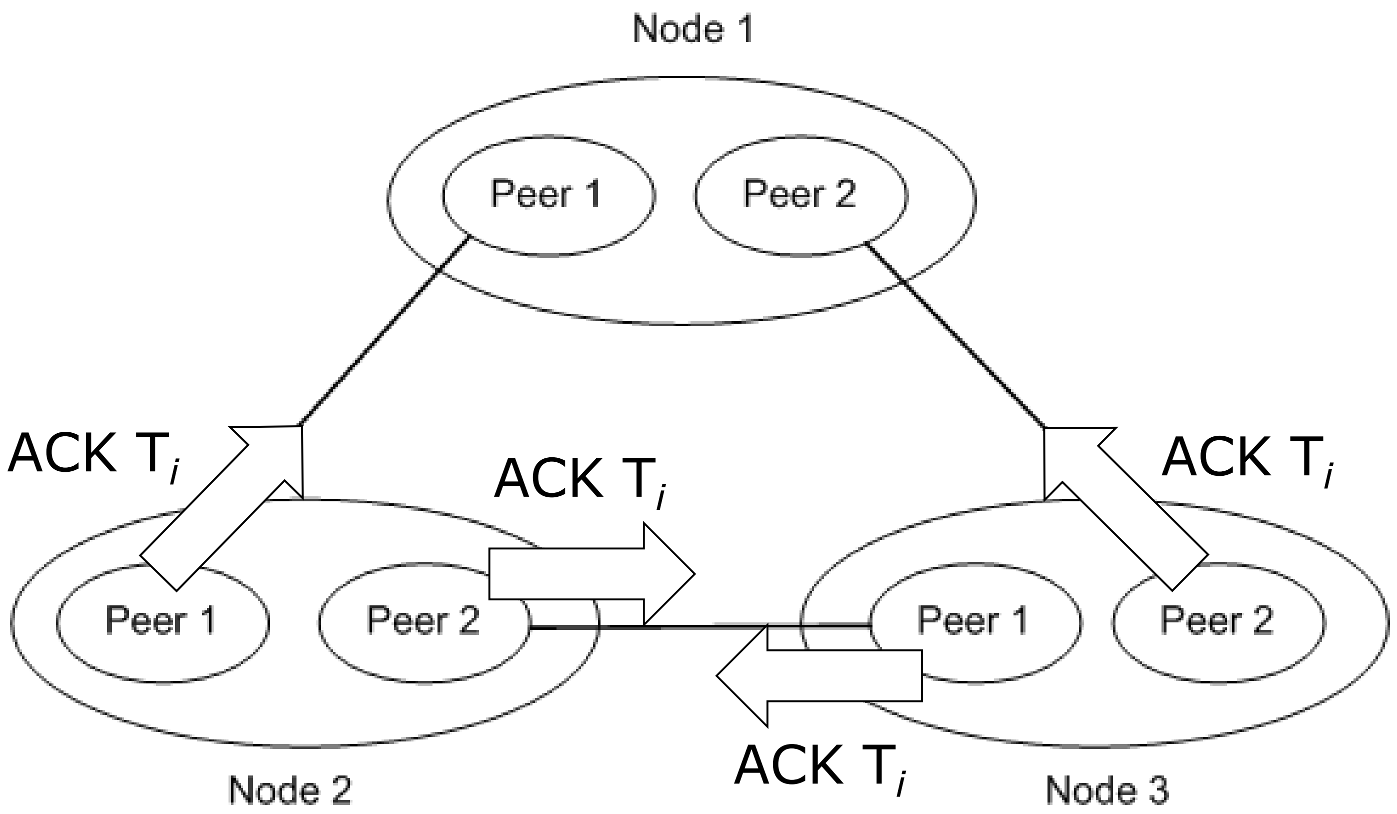
Setting up connections



Multicast & ACK



Multicast & ACK


Node 1
Peer 1
Peer 2
ACK T_i
ACK T_i
ACK T_i
Peer 1
Peer 2
Peer 1
Peer 2
ACK T_i
Node 2
Node 3