```python
#!/usr/bin/python

import socket
import Queue
import threading
import md5
import time

class Entry(object):
    def __init__(self, dynid, sharedSecret, name):
        self.dynid = dynid
        self.sharedSecret = sharedSecret
        self.name = name
        self.lastEventTime = 0
        self.address = ''

class IllegalEventException(Exception):
    def __init__(self, msg):
        self.msg = msg


class Event(object):
    def __init__(self, address, data, receiveTime):
        self.address = address
        self.data = data
        self.receiveTime = receiveTime

    def prepare(self):
        self.port = self.address[1]
        self.address = self.address[0]
        (self.dynid, self.msgTime, self.checksum) = self.data.split(' ')
        self.msgTime = int(self.msgTime)
        self.prepared = True

    def process(self):
        if not self.prepared:
            self.prepare()

        if not ENTRIES.has_key(self.dynid):
            raise IllegalEventException("unknown dynid in event %s" % str(self))
        entry = ENTRIES[self.dynid]

        if self.msgTime + MSG_TIME_CORRIDOR < self.receiveTime:
            raise IllegalEventException("event too old %s" % str(self))
        if self.msgTime - MSG_TIME_CORRIDOR > self.receiveTime:
            raise IllegalEventException("event too young %s" % str(self))


        if entry.lastEventTime >= self.msgTime:
            raise IllegalEventException("timing sequence failure in event, possibly replay %s" % str(self))

        di = "%s %s %d" % (self.dynid, entry.sharedSecret, self.msgTime)
        d = md5.new(di).hexdigest()
        print "%s, received: %s, calculated: %s" % (di, self.checksum, d)
        if d != self.checksum:
            raise IllegalEventException("wrong checksum for event %s" % str(self))

        entry.lastEventTime = self.msgTime

        if entry.address == self.address:
            print "Same address, nothing to do."
        else:
            entry.address = self.address
            print "Set in DNS: %s -> %s" % (entry.name, entry.address)


    def __str__(self):
        if not self.prepared:
            self.prepare()
        return "%s from %s:%d" % (self.data, self.address, self.port)

class Handler(threading.Thread):
    def __init__(self, queue):
        threading.Thread.__init__(self)
        self.q = queue
        self.setDaemon(True)

    def run(self):
        while True:
            event = self.q.get()
            try:
                event.prepare()
                print "Processing event %s" % str(event)
                event.process()
            except IllegalEventException, e:
                print "Some failure: %s" % e.msg
```



```python
class Expirer(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.setDaemon(True)

    def run(self):
        while True:
            print "Expiring ..."
            currentTime = int(time.time())
            for entry in ENTRIES.values():
                if entry.lastEventTime != 0 and entry.lastEventTime + EVENT_LIFE_TIME < currentTime:
                    print "Entry %s expired" % entry.dynid
                    entry.lastEventTime = 0
                    entry.address = NULL_ADDRESS
                    print "Set in DNS: %s -> %s" % (entry.name, entry.address)
            time.sleep(10)


ENTRIES = {
    'testhost': Entry('testhost,', 'test123', 'test.test.de'),
    }
MSG_TIME_CORRIDOR = 5
EVENT_LIFE_TIME = 10
NULL_ADDRESS = '0.0.0.0'

q = Queue.Queue()

handler = Handler(q)
handler.start()

expirer = Expirer()
expirer.start()

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(("", 9090))

while True:
    data, address = s.recvfrom(256)
    try:
        q.put_nowait(Event(address, data, int(time.time())))
    except Queue.Full:
        print "Event %s from %s dropped" % (data, str(address))
```