Subject: Socket: SYN_SENT stalled when ipscan connections in CoWork (MT)
Posted by kohait00 on Thu, 26 Aug 2010 14:36:41 GMT
View Forum Message <> Reply to Message

hey guys,

i am having trouble with Socket (btw: there is hardly any API info for it).

to make a long story short: my app has got some kind of ip scan behaviour to find arbitrary devices spread on the network. i use CoWork for it to do this. but as the scan proceeds (intervall of some 30-40 ip's) the 'netstat -a' lists some of them as SYN_SENT, until it piles up to 10 of those. (i know of XP limit of 10 half open connections). but what surprises me the most: i am correctly closing the Socket actually, when it cant find no connection (ref code borrowed from http server in upp).

this problem only arises when using CoWork and firing a lot of those.. when using my WorkQueue (a single thread thread pool) everything is alright, no SYN_SENT pile. even with CoWork, when i fire only 10 of em, wait till all complete, then fire again, it works fine..

now is this a Upp problem or a windows problem? how can it be that SYN_SENT connections pile up while i normally close the sockets?

a testcase is provided. it is my WorkQueueTest modified. to fire some connecitons leftclick multiple times. ip base is set hardcode 192.168.10.170+X, incrementing each click..

EDIT: since the CoWork has CPU_Cores+2 threads, i have 4 of em. so 4 connections are searched for at once. if one thread finishes, it grabs the next connection intent. so there are almost always 4 pending scans, which is less then the 10 limit. so i am kinda lost here..

as soon as the program is quited though, the stalled SYN_SENT disappear..why does the system maintain those half oopened conns when i am not interested in them (closed it).

this is basicly the code that i am using

```
 if(!socket.IsOpen()) {
  if(!ClientSocket(socket, sock_host, sock_port, true, NULL, 0, false)) {
   error = Socket::GetErrorText();
   socket.Close();
   socket.Clear();
   aborted = true;
  }
  else
   socket.Linger(0);
 }
 if(!aborted)
 while(!socket.PeekWrite(1000)) {
  int time = msecs();
  if(time >= end_time) {
```

```
   error = NFormat(t_("%s:%d: connecting to host timed out"), sock_host, sock_port);
   socket.Close();
   socket.Clear();
   break;
  }
  if(progress(time - start_time, end_time - start_time)) {
   aborted = true;
   error = "Aborted";
   socket.Close();
   socket.Clear();
   break;
  }
 }

...
//some where here the connection can be used, but is open if not aborted, but this is not very important
```

EDIT: WRONG TESTCASE, down there the right one.
