

## Re: Thread deadlock misery

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-03/msg00446.html>

---

- *From:* "PaulH" <[paul.heil@xxxxxxxx](mailto:paul.heil@xxxxxxxx)>
  - *Date:* 13 Mar 2007 10:37:34 -0700
- 

I don't know if this is a 100% reliable source, but according to C++ Report, 11(7), July/August 1999 (<http://www.gotw.ca/publications/mill10.htm>)

All known commercial implementations of vector are contiguous and the "next" (current?) standard guarantees they will be contiguous.

–PaulH

On Mar 13, 11:03 am, "Ben Voigt" <[r...@xxxxxxxxxxxxxxxx](mailto:r...@xxxxxxxxxxxxxxxx)> wrote:

"PaulH" <[paul.h...@xxxxxxxx](mailto:paul.h...@xxxxxxxx)> wrote in message

[news:1173799206.088807.199380@xx](mailto:news:1173799206.088807.199380@xx)

I used the critical section because I read in another post on this group that it might be necessary despite the Interlocked() operation being atomic. In this case it obviously doesn't help me any, so I'll remove it. Thanks!

The Sleep() call in the transmit thread is to regulate the number of transmissions per second. The Sleep() call in the StatCalc thread is to regulate the number of statistical samples per second.

Below is some actual code from my transmit thread. After it sends about 300 frames, this thread halts.

```
/// count of frames sent
```

## Re: Thread deadlock misery

```
volatile long m_nSentFrames;
```

```
////////////////////////////////////  
// Method public static CMainFrame::TransmitThread  
/// @brief Transmit unicast frames to a client at a specified rate.  
///  
/// @param TFP – [in] allocated pointer to the parameters  
/// @return DWORD WINAPI – 0  
////////////////////////////////////  
/*static*/ DWORD WINAPI CMainFrame::TransmitThread( TTCB* TFP )
```

Declare your thread procedure properly, with a void\* parameter, and use a cast inside the function.

Never never never cast a function pointer.

```
{  
    std::auto_ptr< TTCB > pATFP( TFP );
```

You use this pointer constantly, so place all your logic inside a member function of class TTCB.

Then your thread procedure becomes just:

```
{  
    std::auto_ptr<TTCB> pATFP( static_cast<TTCB*>(TFP) ); // might need  
    reinterpret_cast  
    pATFP->RunTransmitLoop();  
}
```

```
WSAData wsaData = { 0 };  
WSAStartup( MAKEWORD( 2, 2 ), &wsaData );
```

If possible, call WSAStartup only once, from your main thread.

```
//  
// Get the address info for the DUT  
//  
struct addrinfo *DUTAddr = NULL,  
hints = { 0 };
```

## Re: Thread deadlock misery

```
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_DGRAM;
hints.ai_protocol = IPPROTO_UDP;
char cPort[ 10 ] = { 0 };
_itoa_s( TFP->port, cPort, 9, 10 );
if( getaddrinfo( pATFP->TFP.address, cPort, &hints, &DUTAddr ) !=
0 )
```

mixing TFP-> with pATFP-> ??? Isn't really a problem, just bad style.

Using a member function will eliminate issues like this.

I guess you are using getaddrinfo instead of filling in a sockaddr\_in yourself in order to handle IPv6? Looks ok.

```
{
// error handling
}

//
// open a UDP, DGRAM socket for the connection
//
SOCKET DUTSocket = socket( AF_INET, SOCK_DGRAM,
IPPROTO_UDP );
if( DUTSocket < 0 )
{
// error handling
}

//
// create an empty buffer to pad the frames to the appropriate
size
//
std::vector< char > buffer( pATFP->TFP.size );
WSABUF wsaBuf = { pATFP->TFP.size, &buffer.front() };
```

I don't know if that's guaranteed to be contiguous memory, better to use:

```
std::auto_ptr buffer = new char[TFP.size];
```

## Re: Thread deadlock misery

```
//  
// Throw unicast frames at the given rate & size to the DUT  
//  
CMainFrame* pParent = reinterpret_cast< CMainFrame* >( pATFP-  
    pParent );  
  
while( pATFP->pParent->m_bRunning )
```

m\_bRunning probably needs to be volatile too

```
{  
    DWORD dwStartTime = GetTickCount();  
    DWORD sent = 0;  
    if( WSASendTo( DUTSocket,  
        &wsaBuf,  
        1,  
        &sent,  
        0,  
        DUTAddr->ai_addr,  
        sizeof( SOCKADDR ),
```

This looks wrong. sizeof(SOCKADDR) is meaningless. Isn't the actual size available in DUTAddr?

```
    NULL,  
    NULL ) < 0 )
```

you're not doing overlapped I/O, so just use BSD-style sendto

```
{  
    // error handling  
}  
InterlockedIncrement( &pParent->m_nSentFrames );  
DWORD dwSleepTime = ( 1000 / pATFP->TFP.FPS ) -  
( GetTickCount() - dwStartTime );  
if( dwSleepTime > 0 )  
    Sleep( dwSleepTime );
```

What happens when GetTickCount() overflows back to zero? Also, you're accounting for the delay in your processing, but not the variation in Sleep times, so your overall transmit rate is lower than you think. Use a waitable timer instead.

Re: Thread deadlock misery

```
}
```

```
//  
// Test finished successfully, cleanup  
//  
closesocket( DUTSocket );  
freeaddrinfo( DUTAddr );  
WSACleanup();  
return 0;  
}
```

If you can tell where I'm going wrong here, please let me know.  
Thanks,  
PaulH