Subject: Re: Kqueue/epoll based interface for TcpSocket and WebSocket
Posted by mirek on Sat, 05 May 2018 15:50:29 GMT
View Forum Message <> Reply to Message

Quote:
Hi, Mirek! Sorry for delay

Well, my delay was worse than yours...

Quote:
Thank you for the answer to this post:
https://www.ultimatepp.org/forums/index.php?t=msg&th=103 17&start=0&
>Why is T_SOCKET a template parameter? Because of websocket?
Yes. The interface is intended to be usable with both tcpsockets and websockets

But not at the same time, right? That would be a big problem IMO...

Quote:
>Vector<SocketEvent<T_SOCKET>> Wait(int timeout); is clumsy - this will IMO cause problems with mapping T_SOCKET back to its "processes".
Could you explain, what kind of problems with mapping will cause the interface?
Let me show, how`Wait(...)` can be used:

```
if(event.IsTriggeredRead())
{
   if(event.GetSocket()->GetSOCKET() == server.GetSOCKET())
   {
                        // new connection
   }
   else
   {
      CLOG("SERVER incoming data: " << event.GetSocket()->GetLine());
```

Now what. Where will these data go?

IMO, there will be a class instance that will provide the communication with single client and you will want to let it read the data. You will probably have some container of these instances. In the end, you need to map incomming event to a class instance somehow. If all ID you have is GetSocket, you will need map socket->instance.

Quote:
Main idea was to serve only 'triggered' sockets without necessity of searching them in array.

Well, I think what I see is quite opposite But I might be wrong.

Quote:
Why `Set(int ii, TcpSocket& s, dword events)`, `Insert(int ii, TcpSocket& s, dword events)` and `Remove(int ii, TcpSocket& s, dword events)` use `TcpSocket& s` but not `SOCKET s` as a parameter?

Sorry, that was my mistake. Actually, the interface between this and actual sockets is something to be carefully considered. With websocket, I have attempted some initial solution with GetWaitEvents and probably AddTo or something like that.

Quote:
Are `Clear(int ii)` and `FindEmpty()` intended to keep user's `Array<***Socket>` and SocketWaitEvent containers synchronized? Does it mean that user can replace one socket in array with another?

Yes.

Quote:
I think prototype for `Remove(...)` should be like this
```// completely remove events attached to socket from event queue (usable for select/kqueue)
// for epoll it will work like 'Disable(int ii, dword events)' (see below) until there are still events that socket is subscribed for
void  Remove(int ii, dword events);```

Another mistake in my proposal, it should have simply been Remove(int ii).

Quote:
and there should be another member function in SocketWaitEvent interface:
```// do not remove event attached to socket from queue but disable notification delivery
void Disable(int ii, dword events);```

Disable would be really hard to use. I am pretty sure that you need some sort of Set with new set of bit flags. Check GetWaitEvents (in HttpRequest and WebSocket).

Quote:
The problem i can see there is that we must to look through the whole `Vector<Tuple<int, dword>> socket` to find indices of triggered sockets after select/epoll/kqueue 'wait for events' system call has returned. It may work slow on large number of sockets. (edited)

So exactly the same problem I can see with your proposal

Anyway, here the idea is that you will maintain that 'int' to be directly mapped to the index of instance of class that handles the communication.

I think here I need to be a little bit more clear about the Clear and FindEmpty - these are meant to make possible to have 'array with holes', in a sense that some indicies are not connected to active instances so that you do not need to do inserts/removes to the array. This might be overkill, as we can make inserts/removes really fast with InArray...

Mirek
