Robotics Service Bus - Feature #41
Implement Message Sequencing

08/18/2010 01:30 AM - S. Wrede

Status:	Resolved	**Start date:**	03/17/2011
Priority:	Normal	**Due date:**	
Assignee:	M. Goetting	**% Done:**	100%
Category:	C++	**Estimated time:**	0.00 hour
Target version:			

Description

In order to send larger event notifications over fragmented port implementations.

Subtasks:

Tasks # 225: Document message format and sequencing strategy	**Resolved**

History

#1 - 08/18/2010 09:58 AM - J. Wienke

- Category set to C++

#2 - 08/24/2010 11:55 AM - S. Wrede

- Due date set to 08/30/2010

- Assignee set to M. Goetting

- Estimated time set to 10.00

#3 - 08/27/2010 01:21 PM - S. Wrede

Important Classes:

- SpreadMessage
- SpreadConnection

First test case could be to just send lena over the wire and see what happens...

#4 - 08/28/2010 05:04 PM - S. Wrede

I just came across an example for msg sequencing in noted once. I'll post it here just for reference, however we're going to implement it:

```
//-----------------------------------------------------------------------------
int vtkSocketCommunicator::SendInternal(int socket, void* data, int length)
{
  char* buffer = reinterpret_cast<char*>(data);
  int total = 0;
  do
    {
    int n = send(socket, buffer+total, length-total, 0);
    if(n < 1)
      {
      return 0;
      }
    total += n;
    } while(total < length);
  return 1;
```

```
}

//-----------------------------------------------------------------------------
int vtkSocketCommunicator::ReceiveInternal(int socket, void* data, int length)
{
  char* buffer = reinterpret_cast<char*>(data);
  int total = 0;
  do
    {
#if defined(_WIN32) && !defined(__CYGWIN__)
    int trys = 0;
#endif
    int n = recv(socket, buffer+total, length-total, 0);
    if(n < 1)
      {
#if defined(_WIN32) && !defined(__CYGWIN__)
      // On long messages, Windows recv sometimes fails with WSAENOBUFS, but
      // will work if you try again.
      int error = WSAGetLastError();
      if ((error == WSAENOBUFS) && (trys++ < 1000))
        {
        Sleep(1);
        continue;
        }
#endif
      return 0;
      }
    total += n;
    } while(total < length);
  return 1;
}

//-----------------------------------------------------------------------------
int vtkSocketCommunicator::SendTagged(void* data, int wordSize,
                                      int numWords, int tag,
                                      const char* logName)
{
  if(!this->SendInternal(this->Socket, &tag, static_cast<int>(sizeof(int))))
    {
    if (this->ReportErrors)
      {
      vtkErrorMacro("Could not send tag.");
      }
    return 0;
    }
  int length = wordSize * numWords;
  if(!this->SendInternal(this->Socket, &length,
      static_cast<int>(sizeof(int))))
    {
    if (this->ReportErrors)
      {
      vtkErrorMacro("Could not send length.");
      }
```

```
    return 0;
    }
  if(!this->SendInternal(this->Socket, data, wordSize*numWords))
    {
    if (this->ReportErrors)
      {
      vtkErrorMacro("Could not send message.");
      }
    return 0;
    }

  // Log this event.
  this->LogTagged("Sent", data, wordSize, numWords, tag, logName);

  return 1;
}

//-----------------------------------------------------------------------------
int vtkSocketCommunicator::ReceiveTagged(void* data, int wordSize,
                                         int numWords, int tag,
                                         const char* logName)
{
  int success = 0;
  int length = -1;
  while ( !success )
    {
    int recvTag = -1;
    length = -1;
    if(!this->ReceiveInternal(this->Socket, &recvTag,
        static_cast<int>(sizeof(int))))
      {
      if (this->ReportErrors)
        {
        vtkErrorMacro("Could not receive tag. " << tag);
        }
      return 0;
      }
    if(this->SwapBytesInReceivedData == vtkSocketCommunicator::SwapOn)
      {
      vtkSwap4(reinterpret_cast<char*>(&recvTag));
      }
    if(!this->ReceiveInternal(this->Socket, &length,
        static_cast<int>(sizeof(int))))
      {
      if (this->ReportErrors)
        {
        vtkErrorMacro("Could not receive length.");
        }
      return 0;
      }
    if(this->SwapBytesInReceivedData == vtkSocketCommunicator::SwapOn)
      {
      vtkSwap4(reinterpret_cast<char*>(&length));
```

```
      }
    if(recvTag != tag)
      {
      char* idata = new char[length + sizeof(recvTag) + sizeof(length)];
      char* ptr = idata;
      memcpy(ptr, (void*)&recvTag, sizeof(recvTag));
      ptr += sizeof(recvTag);
      memcpy(ptr, (void*)&length, sizeof(length));
      ptr += sizeof(length);
      this->ReceivePartialTagged(ptr, 1, length, tag, "Wrong tag");
      int res = this->InvokeEvent(vtkCommand::WrongTagEvent, idata);
      delete [] idata;
      if ( res )
        {
        continue;
        }

      if (this->ReportErrors)
        {
        vtkErrorMacro("Tag mismatch: got " << recvTag << ", expecting " << tag
                      << ".");
        }
      return 0;
      }
    else
      {
      success = 1;
      }
    }
  // Length may not be correct for the first message sent as an
  // endian handshake because the SwapBytesInReceivedData flag
  // is not initialized at this point.  We could just initialize it
  // here, but what is the point.
  if ((wordSize * numWords) != length &&
      this->SwapBytesInReceivedData != vtkSocketCommunicator::SwapNotSet)
    {
    if (this->ReportErrors)
      {
      vtkErrorMacro("Requested size (" << (wordSize * numWords)
                    << ") is different than the size that was sent (" << length << ")");
      }
    return 0;
    }
  return this->ReceivePartialTagged(data, wordSize, numWords, tag, logName);
}

//-----------------------------------------------------------------------------
int vtkSocketCommunicator::ReceivePartialTagged(void* data, int wordSize,
                                                int numWords, int tag,
                                                const char* logName)
{
  if(!this->ReceiveInternal(this->Socket, data, wordSize*numWords))
    {
```

```
    if (this->ReportErrors)
      {
      vtkErrorMacro("Could not receive message.");
      }
    return 0;
    }
  // Unless we're dealing with chars, then check byte ordering.
  // This is really bad and should probably use some enum for types
  if(this->SwapBytesInReceivedData == vtkSocketCommunicator::SwapOn)
    {
    if(wordSize == 4)
      {
      vtkDebugMacro(<< " swapping 4 range, size = " << wordSize
                    << " length = " << numWords);
      vtkSwap4Range(reinterpret_cast<char*>(data), numWords);
      }
    else if(wordSize == 8)
      {
      vtkDebugMacro(<< " swapping 8 range, size = " << wordSize
                    << " length = " << numWords );
      vtkSwap8Range(reinterpret_cast<char*>(data), numWords);
      }
    }

  // Log this event.
  this->LogTagged("Received", data, wordSize, numWords, tag, logName);

  return 1;
}

//-----------------------------------------------------------------------------
template <class T, class OutType>
void vtkSocketCommunicatorLogArray(ostream& os, T* array, int length, int max,
                                   OutType*)
{
  if(length > 0)
    {
    int num = (length <= max)? length:max;
    os << " data={" << static_cast<OutType>(array[0]);
    for(int i=1; i < num; ++i)
      {
      os << " " << static_cast<OutType>(array[i]);
      }
    if(length > max)
      {
      os << " ...";
      }
    os << "}";
    }
```

#5 - 08/28/2010 05:08 PM - S. Wrede

From previous considerations about this topic (just ideas from earlier times):
How to encapuslate message de-multiplexing?

- Write basic SpreadNetworkMessage class that serializes data plus status information on a stream, at least * if message is self-contained * how many parts follow * number of current part * if it is the final part * what else?
- Extend SpreadMessage class by * isComplete() * isError()
- An incremental buildup factory method for construction of SpreadMessage out of a sequence of SpreadNetworkMessages?!?
- The semantics of SpreadConnection::receive would change as it shall provide the next 'complete' message to its caller

Just ideas from ancient brainstorming... ;-)

#6 - 12/04/2010 01:25 PM - S. Wrede

Any updates on the status? We probably need this feature now for HUMAVIPS... So, lets discuss hwo to solve this on Monday!

#7 - 01/15/2011 04:22 PM - S. Wrede

Any updates on this topic?

#8 - 01/17/2011 03:02 PM - S. Wrede

- Status changed from New to Feedback
- Priority changed from High to Urgent

#9 - 03/08/2011 09:25 PM - J. Wienke

We reaaaallllllyyyyyyyyyy need this for the humavips demonstrator to not rely on the strange patched spread daemon anymore. Any chance that you can implement this during the next days?

#10 - 04/29/2011 05:50 PM - J. Wienke

Isn't this implemented now?? Why is the ticket still opened?

#11 - 05/04/2011 01:07 PM - J. Wienke

- Status changed from Feedback to Resolved

There even is a unit test now. It works ;)