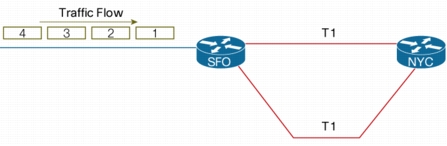
**Reliable Messaging**

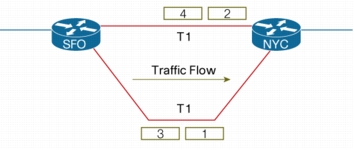
**Problem:**

Internet is vast with many different possible connections between two nodes. With the advent of applications getting integrated frequently, achieving reliability over HTTP transport protocol by the heterogeneous applications is difficult. The inherent lack of transport reliability of HTTP is very problematic in such scenarios for two reasons

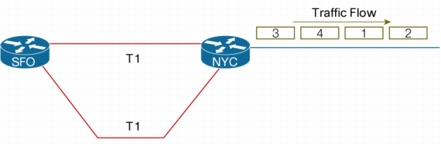
* Guarantee delivery
* Sequence of deliver.



**Figure 1:** Messages sent in order from Sender



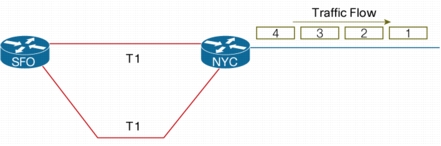
**Figure 2:** Messages traversing through different network



**Figure 3:** Packets arriving out of sequence at Receiver

**Solution:**

The transport-independent WS-Reliable Messaging (or WS-RM) protocol allows for creating reliable communication paths over unreliable connections and protocols. It does so by establishing end-to-end communication sessions and by introducing explicit acknowledgement of messages into the communication flow.



**Figure 4:** After reliable messaging, messages arriving in sequence at Receiver

**Working:**

A client sends a sequence of messages across a communication link and asks the receiver to acknowledge that it has received the message(s).

The acknowledgement(s) are sent back to the client and once the client has received the acknowledgement, it knows that the message(s) has been successfully transferred.

WS-Reliable Messaging works very similar to TCP protocol but at a layer above transport. It independent of the underlying transport. More concretely: WS-RM is designed to control reliable delivery of single SOAP messages or sequences of SOAP messages between two endpoints, irrespective of how these endpoints are connected; for instance even if messages travel through message routers or other intermediaries using different transport protocols for each of hop.

To establish a reliable messaging link between two endpoints, we first need a notion of a connection or "session". Each message sent in the context of that session is assigned a unique number denoting its order in the sequence. The sender-side (or "initiator") establishes a temporary cache to keep track of the messages sent and matches them up with the incoming acknowledgements. If a message is not acknowledged after a certain time (the "retry-timeout"), the message is resent automatically from the cache. Once an acknowledgement is received, the message can be removed from the cache.

The receiver-side (or "acceptor") establishes a cache for holding messages it accepted in order to store them before delivering them to the application. That's done because the receiving infrastructure and the application may have a "pull-relationship".

**References:**

<http://msdn.microsoft.com/en-us/library/aa480191.aspx>

<http://msdn.microsoft.com/en-us/library/hh128101(v=prot.13).aspx>

<http://msdn.microsoft.com/en-us/library/cc219293.aspx>