Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on October 4, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This paper shows that the currently defined per-message hop-by-hop mechanism doesn't scale for large deployments and is not feasible for scalable IM conferencing. Instead, the paper proposes to replace this mechanism with an MSRP layer "per TCP connection keep-alive mechanism". This paper also reinforces the need for the end-to-end mechanism to be defined in the core MSRP specification and to become the abstraction for all MSRP applications regardless the underlying topology in use.
# Table of Contents

1. Conventions and Terminology ............................................. 3  
2. Introduction .......................................................... 3  
3. Current Architecture .................................................. 3  
   3.1 Hop-by-Hop Mechanism .............................................. 4  
   3.2 End-to-End Mechanism ............................................. 4  
4. Scalability Analysis ................................................... 5  
   4.1 Message State Maintenance ....................................... 5  
   4.2 Simplistic TCP over TCP ......................................... 6  
   4.3 MSRP Conference Focus ........................................... 6  
   4.4 Benchmark ........................................................ 7  
5. Functional Overview ................................................... 8  
   5.1 Hop-by-Hop .......................................................... 9  
      5.1.1 Achieved Functionality .................................... 9  
      5.1.2 Functionality that is NOT Provided ....................... 9  
   5.2 End-to-End .......................................................... 10  
      5.2.1 MSRP Abstraction ........................................... 10  
      5.2.2 Requirements ................................................ 10  
6. Proposed Architecture ................................................ 11  
   6.1 Keep-Alive Mechanism per TCP Connection ....................... 11  
   6.2 Modular End-to-End Mechanism .................................. 12  
7. Conclusion .................................................................. 13  
8. Acknowledgments ......................................................... 13  
9. References ................................................................ 13  
   9.1 Normative References .............................................. 13  
   9.2 Informational References ......................................... 14  
      Author’s Address .................................................... 14  
      Intellectual Property and Copyright Statements .............. 15
1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

2. Introduction

Message Session Relay Protocol (MSRP) [3] is designed to transfer Instant Messages (IM) over a reliable transport protocol (e.g. TCP). Currently two acknowledge mechanisms are being designed for MSRP: the mandatory hop-by-hop mechanism (in addition to the underlying TCP) and the optional end-to-end mechanism.

This paper argues that the defined hop-by-hop transaction statefull per message mechanism provides a small number of advantages while it creates unnecessary system and network overload of ~20%. Moreover, this mechanism is hardly applicable for IM conferencing.

In order to illustrate these arguments, this paper examines the CPU performance impact of the hop-by-hop acknowledge algorithm on different IM topologies.

As a conclusion, this paper proposes to modify the hop-by-hop mechanism to become per-TCP connection (instead of per instant message) and to position the end-to-end mechanism as the IM abstraction for all MSRP applications regardless the underlying topology in use.

3. Current Architecture

This section provides a summary of the two currently proposed mechanisms: hop-by-hop and end-to-end. The intention is that the two mechanisms complement each other in order to meet the MSRP requirements.

The hop-by-hop mechanism (sometimes referred to as the "ACCEPT" mechanism) is modeled after the SIP transaction model. The MSRP hop-by-hop mechanism is transaction statefull per message and is used for immediate positives acknowledgments from the next hop only. (Negative responses are also defined but they are applicable only for deployments with no relays in the network.) The hop-by-hop mechanism is mandatory for both relays and user agents.

The negative "end-to-end" mechanism is mandatory for relays for error reporting when connectivity problem is detected. In this context the mechanism is interchangeably referred to as "DSR" or "DSN". Relays MUST NOT use the end-to-end mechanism for reporting positive
acknowledgements.

The end-to-end mechanism is also envisioned to be optionally used by end user applications for both positive and negative acknowledgements. In this context the mechanism is usually referred to as the "MDN" mechanism.

3.1 Hop-by-Hop Mechanism

When an MSRP entity A (a user agent or a relay) sends/forwards a message, it creates the message transaction state and starts a timer (~30 sec). Once the message is received at the next hop B (which can be a user agent or an MSRP relay), the MSRP entity (B) MUST send a positive acknowledgement to the previous hop (A). If there is an immediate problem in the local handling of this message by B, the MSRP entity (B) MUST generate an error report and send it all the way back to the originator (i.e. the original source) of the message using the end-to-end mechanism described below.

Once A receives a positive acknowledgement for the message, it cleans the state for this message (including the timer).

Otherwise, if the timer for the non-acknowledged message expires, a relay (e.g. A) creates a negative error report using the end-to-end mechanism described below.

As a result, all MSRP entities (both user agents and relays) keep transaction states for all (or last "n") messages that have been sent/forwarded by the entity but neither positive acknowledgements have been received for them nor their application timers have expired.

If at certain point of time a problem in the outgoing TCP connection is locally detected, the MSRP relay MUST create and send back error reports individually to each message originator using the end-to-end mechanism described below. In the same event, the user agent MUST report the failure to the local application(s). In both cases, the error reports must include a list the messages per originator that havenÂ’t been acknowledged by the next hop.

3.2 End-to-End Mechanism

The baseline design of the end-to-end mechanism assumes the following:

- When an MSRP entity (relay or user agent) detects a TCP local failure or a failure in a message delivery either to the next hop or to the local application, the entity creates an error report
that includes the ID or IDs of the message(s) in problem

- The entity MUST have enough information in order to send the error report all the way back to the originator of the message(s) in problem INSIDE the existing MSRP connection, which the original message(s) belonged to.

- In addition, if as a result of the SDP negotiation, a user agent agreed to provide an application layer positive and/or negative report/receipt per message, the user agent MUST create the report and send it back to the message originator in the same manner as described above (i.e. inside the existing MSRP connection).

The following known issues still need to be addressed by the MSRP specification:

- Relationship between the MSRP end-to-end procedures and the "application above the user agent" end-to-end procedures
- The exact end-to-end operation modes and their negotiation using the offer-answer SDP mechanism
- The exact format of the reports (both negative and positive)
- A mechanism needs to be defined so that all outstanding end-to-end reports (i.e. belonging to sessions ended earlier by user agents) are silently discarded by the system.
- A mechanism needs to be defined so that NO end-to-end status reports will be generated about delivery of earlier end-to-end status reports.
- It is unclear why the status report messages need to be acknowledged hop-by-hop (as per current design) because they MUST not be included in the status reports (as stated above).

4. Scalability Analysis

This section shows how the currently defined MSRP hop-by-hop mechanism (on the top of the existing TCP connections) affects the overall MSRP system scalability.

4.1 Message State Maintenance

The most trivial consequence of the defined "hop-by-hop" mechanism is the need to keep per message a state object (including a timer) and to clear it upon receiving an acknowledgement for the message.

In order to fully realize the impact of this design on an IM system, we need to translate it to numbers. For example, as of today, a typical public IM service would have 150MM subscribed users with a peak of 10MM simultaneous users.

Consequently, in order to compare the cost of such IM solution to "less statefull" IM architectures, every additional bit of memory and
CPU processing power required for keeping the state per "message transaction" need to be multiplied at least (since message transactions can be overlapped) by 10MM.

It is interesting to mention that even modern TCP implementations have moved away from acknowledging every TCP segment and today acknowledge every second segment only.

4.2 Simplistic TCP over TCP

One of the concerns with the suggested hop-by-hop mechanism is that although the mechanism is very simple, it suffers from some of the known TCP problems but, unlike TCP, doesn't address them. Unfortunately, the current working assumption that the "congestion-safe" TCP will automatically make the MSRP congestion-safe is inaccurate.

For example, in order to overcome the well known FW/NAT traversal problems, both Responses and Requests (instant messages) between the users will be sent over the same TCP connection.

Let's assume that users A and B have an IM session between them over a single TCP connection.

Because each message will be acknowledged (by Response) by B automatically, A needs to control the rate of the messages it sends to B in order to be able to process the incoming responses. In order to do so, A buffers messages before it sends them to B. In the same time B sends its data messages (i.e. new Requests) asynchronously on the same connection from B to A. As a result, the connection gets throttled by new Requests while the old Requests (from A to B) haven’t been acknowledged yet. Provided A and B do not use common throttling mechanism and/or have different processing power this situation requires careful tuning (e.g. configuration or negotiation) by each side.

The situation becomes much more complex when more than a single IM conversation is multiplexed over a single TCP connection.

The situation becomes practically unmanageable in case of IM conference focus as described below.

4.3 MSRP Conference Focus

An IM conference, a focus or a mixer of n participants, upon receiving an instant message from one of the participants, needs to respond to it and send (n-1) copies to all other participants (or multicast a single copy to all). The focus/mixer needs to sustain
processing of (n-1) responses being sent back to it per each single message in the conference. Since a number of participants in a conference can change dynamically, so will need the throttling parameters of the IM conference implementation!

4.4 Benchmark

The results shown below reflect a typical setup which was not optimized to prove the case but rather shows typical affect of instant messages being acknowledged by the receiver.

The presented benchmark doesn't take into consideration error probability and the overhead of sending error reports to the message originator.

The setup consists of a SIP application server and a simulator of 20000 user agents. All SIP connections run over TLS. No additional proxies or relays are involved.

In this benchmark the applications use SIP MESSAGE method inside SIP (INVITE) dialog to deliver the instant messages. The CPU load is shown in % which makes the results applicable to any other format (e.g. light MSRP) where the requests and the responses processing is proportional to that of SIP.

The performance statistics was measured on a SIP Server that acts as a Registrar to 20000 SIP user agents. The server also serves as an IM focus for each user. Each user periodically (in constant rate) establishes IM conference with 49 other users, exchanges SIP MESSAGEs with the conference focus which funnels the MESSAGE to other 49 participants.

"Normal run" column represents SIP compliant behavior with every MESSAGE being acknowledged by response. The Server keeps state information per message and updates it upon receiving the responses.

"Drop response" column represents a run where user agents generate responses to every MESSAGE they receive. The Server parses the responses on TCP (TLS) and SIP layers but doesn't keep state information per message and doesn't apply the information to the application logic.

"No response" column represents a run where user agents don't generate responses to MESSAGEs they receive. The Server doesn't keep state information per message and doesn't receive the responses to process.
The following observations can be deduced from the results shown in the table:

- Server's CPU is not running at 100% usage. Therefore, the throttling problem described above does not affect the statistics results.
- We can see a drop of 10% CPU between the runs (i.e. ~76%, ~65%, and ~55%).
- CPU processing power increases by 20% in order to process the responses in this typical (far from extreme) scenario.

5. Functional Overview

In order to find a better alternative to the current hop-by-hop mechanism and in the same time to meet all critical IM requirements,
this section examines each of the mechanisms in terms of the requirements they address and incapable to address.

5.1 Hop-by-Hop

5.1.1 Achieved Functionality

The defined hop-by-hop mechanism runs over TCP. Therefore, in addition to the functionality provided by TCP, the mechanism adds the following two properties:

Detection of TCP connection failure earlier than reported by TCP itself:

- This can be useful if TCP connection shows a property of unreasonably long (e.g. up to 9 minutes) TCP retry procedures. The reason for this pattern would be either improperly tuned TCP configuration or/and transport layer intermediates (potentially deployed for security or monitoring) that actively intervene the TCP connection and, as a result, change its properties
- Unfortunately, using the MSRP timer with shorter than TCP timeout will increase the probability of message duplications as described in the next section

Ability to indicate specific messages (that with high probability didn’t get delivered to the destination) in the "end-to-end" error reports generated by relays:

- This functionality is achieved by keeping the state with the message information (i.e. message ID and optionally more details) in the relays for all the messages that haven’t been positively acknowledged by the next hop

5.1.2 Functionality that is NOT Provided

MSRP POSITIVE acknowledgement from the next hop does NOT mean that the message was actually delivered to its destination:

- Although the delivery to the next hop was successful, the message delivery can potentially fail on any of the subsequent hops towards the destination

"End-to-end" ERROR report (with message IDs) being generated by a relay as a result of hop-by-hop mechanism SHOULD NOT be used for application retransmission logic:

- It’s possible that an MSRP timer of the relay has expired or a local TCP connection failure has been detected by the relay when
there are still locally unprocessed positive acknowledgments from the next hop in the TCP pipe. These acknowledgements can be to the messages that had been already forwarded by the next hop and eventually got delivered to the destination

o Note that as shorter the local MSRP timer is, as more frequently the scenario above can occur

o Note also that the scenario above can theoretically happen to any or all of the relays for the same message. This will result in a flood of error reports towards the message originator even in the case of successful message delivery to its final destination

"End-to-end" ERROR report (with message IDs) being generated by a relay as a result of hop-by-hop mechanism MUST NOT be used as an input to applications that require every message being delivered to the destination once only (i.e. the "no message duplication" requirement for stock traders).

o Same reasons as above

5.2 End-to-End

5.2.1 MSRP Abstraction

As shown above, the defined hop-by-hop mechanism is incapable of addressing most of the basic IM applications requirements when more than a single hop is involved.

In the same time, any proper protocol design MUST shield the underlying network topology from the applications building on the top of this protocol. Applying this concept to the MSRP architecture, it means that IM applications MUST be exposed to the same level of abstraction (i.e. MUST be able to use the same MSRP primitives) regardless whether MSRP relays exist in the network or MSRP runs over a single TCP hop only.

As a result of the two statements above, this paper argues that the end-to-end mechanism MUST be specified in the basic MSRP specification:

o This mechanism will provide the desired network abstraction to be used by IM applications even when no relays are deployed in the network

o This mechanism will provide the migration path towards deployments with relays without affecting the IM applications

5.2.2 Requirements

This paper argues that with a flexible modular design, the end-to-end
mechanism will provide the answer to a broad range of IM application.

This paper envisions that the majority of IM applications will directly use the defined standard primitives. Examples of such applications include:

- Loose IM session: all user messages are sent without receipt, but the application sends a heartbeat message with requesting a receipt for each
- Typical IM session: For all user messages a receipt is requested but the indication messages (e.g. "The user is typing") are delivered without any receipt

Using the defined standard primitives is also SHOULD be possible to build more advanced or and more strict IM applications:

- Some IM application will choose to implement an acknowledge algorithm for "n" last messages in a window. This can be achieved by sending the messages without requesting the MSRP receipt per message, but instead by implementing the application logic with strict sequential IDs for messages
- Some IM applications (e.g. for stock transactions) will require that no messages are duplicated during a SIP session. In order to meet this requirement, even a more sophisticated application can be build using the basic MSRP primitives. In this case, upon failure detection in the MSRP logical connection by one of the user agents, the application will probably establish a new MSRP session inside the same SIP session (e.g. by using re-INVITE). The new MSRP session will trigger the establishment of a new TCP connecton(s). The application will need to define the means for association between the two MSRP sessions and thus will be able to learn the exact status of previously sent messages using the new MSRP session.

6. Proposed Architecture

This section presents a proposal for an alternative MSRP architecture that both scales for large deployments and defines a modular set of basic message primitives thus addressing requirements of a broad range of different IM applications (as described above).

6.1 Keep-Alive Mechanism per TCP Connection

In order to solve the TCP marginal behavior problem, MSRP specification will define an MSRP level single-hop keep-alive mechanism. This mechanism will detect a TCP connection problem in a reasonable period of time (e.g. 10 sec).
Each MSRP entity (both a user agent and a relay) will implement the defined keep-alive mechanism for every TCP connection that is used for MSRP session(s). A single instance of the keep-alive mechanism will be applied for a TCP connection even if a number of MSRP sessions are multiplexed over this TCP connection.

No additional hop-by-hop mechanisms will be used by MSRP.

The error "end-to-end" reports will be created by MSRP relays and will be sent back inside existing MSRP sessions (as before) for all MSRP sessions that potentially got affected by the detected TCP failure.

User Agents will convey the error reports (both received from the network and locally generated) to the IM applications as a part of the MSRP "end-to-end" standard abstraction.

The relays will keep enough information in order to perform this reverse "switching/routing" logic. In reality, in majority of the cases (i.e. for bidirectional MSRP sessions) this information will be kept by the MSRP entities anyway for routing data messages in the opposite direction.

Note that the achieved functionality is only "inferior" to the currently defined hop-by-hop mechanism in a sense that the MSRP error reports will never carry the ID of specific message(s) belonging to the MSRP session.

In fact, as this paper showed earlier, the message IDs carried in the error reports generated by relays are useless for application logic anyway because this information is not accurate and can simply be misleading.

Instead, IM applications that really need to know the status of specific or all sent messages will use the end-to-end mechanism described below.

### 6.2 Modular End-to-End Mechanism

The end-to-end MSRP mechanism will define the following basic primitives PER MESSAGE:

- Send a message unreliably within an MSRP session with no receipt at all
- Send a message within an MSRP session and send back error receipt(s) only
- Send a message within an MSRP session and send back a receipt (either positive or negative)
The following basic procedure SHOULD be defined:

- A heartbeat end-to-end mechanism with configurable timeout within a session

When required, these primitives will allow for building more sophisticated logic on application level.

7. Conclusion

This paper showed that the currently defined per-message hop-by-hop mechanism doesn’t scale for large deployments and is not feasible for scalable IM conferencing, thus missing much of the purpose of a session-mode IM protocol.

Instead, the paper proposes to replace this mechanism with an MSRP layer "per TCP connection keep-alive mechanism" that meets the TCP failure detection requirement and doesn’t create the "per-message" scaling problem. The TCP connection error reports are to be reported to MSRP applications as a part of a broader end-to-end mechanism.

This paper reinforces the need for the end-to-end mechanism to be defined in the core MSRP specification and to become the abstraction for all MSRP applications regardless the underlying topology in use.

8. Acknowledgments

I would like to thank Ben Cambell, Cullen Jennings, Robert Sparks, and Adam Roach for our spirited discussions and for urging me to write this paper.

9. References

9.1 Normative References


9.2 Informational References

Author’s Address

Orit Levin
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

EMail: oritl@microsoft.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF’s procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.