

A RELIABLE TRANSPORT PROTOCOL FOR THE TACTICAL INTERNET

Jay W. Strater

The MITRE Corporation
Battlefield Systems Division
Eatontown, New Jersey

ABSTRACT

This paper presents a candidate transport protocol that is being developed to provide reliable data transfer over tactical communications systems. The paper addresses data transport requirements, protocol comparisons, and baseline protocol modifications. The modifications include improvements in communications efficiency for large or small messages and communications reliability within a tactical internet. The baseline protocol selection is the block transfer protocol NETBLT (RFC 998).

INTRODUCTION

Tactical battlefield communications systems must operate on mobile platforms with constrained bandwidth channels and high interference conditions. Tactical communications performance is often characterized by long and variable delays with high and variable error rates. The Army's Tactical Internet (TI) is a good example of this environment.

To provide reliable data delivery in the TI, several protocols have been considered but each performs unsatisfactorily in TI conditions. The choices include TCP, UDP with end-to-end message retransmission, and Mil Std 2045-14502-1A transport layer (referred to as the S/R protocol).

TCP provides poor performance for short messages because of call setup and termination overhead. It provides poor performance for long messages because it conducts congestion control for all packet losses, queuing or error related. UDP with message retransmission provides unacceptable performance for large packets because it requires full message retransmission for any partial message error. UDP also restricts maximum message sizes to 64 Kbytes. The S/R protocol provides no flow control.

Figure 1 illustrates the representative characteristics of TCP and UDP with message retransmission for one

System Improvement Program (SIP) SINCGARS network within the TI. The results assume a message of 8 512-byte IP packets and a network path over two SINCGARS links. Network access and data link control is provided by Mil Std 188-220A and assumes 20 second (nominal) access delays per link. Two 188-220A types of operation, 1 and 4, are evaluated with TCP and UDP. Type 1 has no automatic repeat request (ARQ) and Type 4 has connectionless ARQ (one ACK for each received packet). Up to 2 retransmissions are assumed for Type 4 links and UDP end-to-end message retransmission. The results are derived analytically and are approximate.

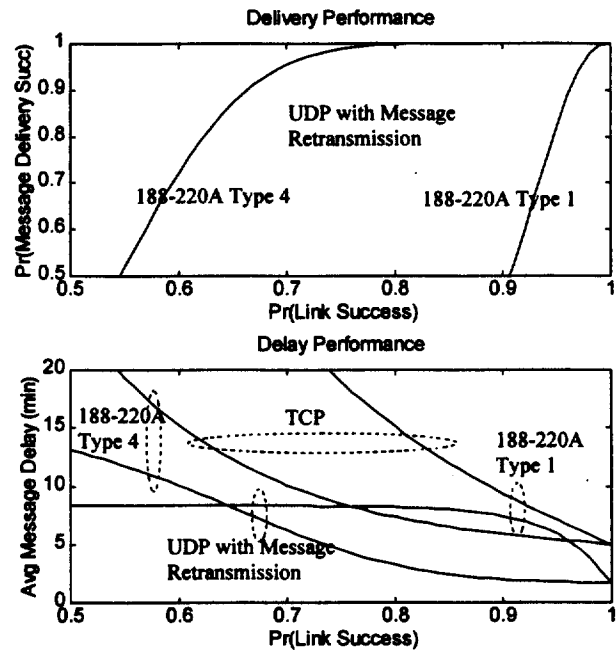


Figure 1. Example SIP Net Performance

The results indicate long delays for all options when link success probability is below 0.9 (common) and message delivery probability is close to 1.¹

To provide reliable and efficient message transfer over tactical communications links, a transport protocol must

¹ TCP has a message delivery probability of 1 for all link success probabilities.

accommodate highly variable delay and delivery probabilities. In addition, a transport protocol must provide efficient operation for arbitrary message sizes. This paper presents a candidate transport protocol that is being developed for reliable data transfer in the TI. The paper addresses data transport requirements, baseline protocol comparisons, and baseline protocol modifications.

The paper is organized in 5 sections. Following the introduction, Section 2 addresses the protocol objectives and requirements. Section 3 presents the results of the protocol assessment. Section 4 addresses the modifications needed for the baseline protocol to satisfy all protocol requirements. Section 5 presents a summary and the next development steps.

OBJECTIVES AND REQUIREMENTS

The following objectives were established at the start of the protocol development:

1. Develop a protocol that provides reliable and efficient communications for tactical conditions, particularly the TI.
2. Use existing protocols as a foundation for the development, making changes only when necessary.
3. Keep the application interfaces simple, making it a standard stream-socket application programmer interface (API).

Requirements were developed for reliable message delivery, flow control, message size scalability, and efficient TI performance to achieve the first objective. A preexisting protocol with software implementations, NETBLT [1], was chosen as the baseline for the development (Section 3) to achieve the second objective. NETBLT code from the MilStd 2045-44500, Tactical Communications Protocol 2 (TACO2) [2], is being modified to provide a standard stream socket API to achieve the final objective. The remainder of this report focuses on the first two objectives, starting with the requirements.

The requirements for reliable message delivery are:

- Guarantee message delivery that is error free with in sequence packet delivery without packet duplication².

² Packets here refer to IP packets that are segmented from the message by the transport protocol into the maximum sizes as allowed by the channel interface.

- Allow timers to adjust to dynamic delay/congestion conditions.
- Accommodate varying channel error conditions.

The requirements for flow control are:

- Limit the number of uncontrolled packets in burst transmissions.
- Match sender and receiver processing rates and match communication link capacities when possible.
- Adaptively adjust transfer rates for congestion control.

The requirement for scalability is to support unlimited message sizes. The requirements for efficient tactical operation are:

- Minimize data packet header size for small packet sizes.
- Provide efficient call establishment and termination for small messages.
- Provide selective packet retransmission, limited acknowledgment (ACK)-to-data packet ratio, and allowance for multiple outstanding unacknowledged transmissions for large messages.
- Provide reliable multicasting.

PROTOCOL COMPARISON

A qualitative assessment of TCP, UDP with message retransmission, the S/R protocol, NETBLT, and VMTP [3] was conducted to choose a preexisting protocol as the bases for TI transport protocol development. TCP and NETBLT were found to satisfy more protocol requirements than the other choices. In general TCP was found to satisfy the most reliability requirements and NETBLT was found to satisfy the most efficiency requirements.

TCP and NETBLT guarantee error free, non-duplicated packet delivery and have adaptive timer calculations. TCP also provides in-order packet delivery and robustness against potential internet problems (from old-lived IP packets or simultaneous command operations) and NETBLT does not. Neither protocol provides reliable multicasting although NETBLT may be easier to modify to support reliable multicasting because it is receiver oriented.³

³ Reliable multicasting has not been addressed yet and is the subject of future efforts.

TCP and NETBLT have significant connection overhead and sizable packet headers.⁴ NETBLT provides selective retransmission and TCP does not in current implementations.⁵ NETBLT also has a low ACK-to-packet ratio whereas TCP sends 1 ACK for every 2 data packets in most implementations. Both protocols allow multiple outstanding packets to provide efficiency when channel delays are large.

TCP and NETBLT limit the size of burst transmissions, have pre-established link capacity restrictions, provide adaptive flow control, and allow effectively unlimited message sizes. TCP provides flow control by a common flow/error control window that advances based on received ACKs. NETBLT provides flow control by a dynamic rate control adjustment that is independent of error control.

NETBLT was selected as the baseline transport protocol because it satisfies the most requirements, can relatively easily be adapted to meet unsatisfied requirements, and has a well used reference implementation available.⁶ NETBLT was deemed to have a significant TI performance advantage over TCP because of its decoupled error and flow control approach and low ACK-to-packet ratio.

NETBLT's feature of decoupling error and flow control, allow it to "flywheel" through temporary error events without inefficiently curtailing information flow. This feature must be carefully designed for shared network (internet) conditions, however, to ensure that congestion control is adequately responsive when congestion and errors cause packet loss. Subsequent evaluations are needed to determine the best settings.

NETBLT sends messages in only one direction (sender to receiver) and sends control message in both directions. Either sender or receiver can initiate connection establishment (becoming the active opener versus the passive opener). NETBLT conducts error and flow control from the receiver and operates by sending aggregated control messages after receiving a buffer of

⁴ Transaction-oriented TCP (T/TCP), however, could be used instead of TCP to limit connection overhead.

⁵ New implementations of TCP, however, are being developed, with selective retransmission.

⁶ The NETBLT implementation, however, does require standard stream-socket API development in addition to internal protocol modifications.

multiple packets. Receiver control provides the advantage of reducing sensitivity to network delays.

PROTOCOL MODIFICATIONS

To satisfy the protocol requirements, NETBLT must be modified to provide the following:

1. Efficient operation for small messages in which connection/termination overhead is reduced.
2. Robust internet operation for standard message transfer to ensure that packets from prior connections can not interfere with connection establishment, data transfer, and termination and simultaneous connection and termination attempts can be resolved.
3. Modified packets and headers to support the first two modifications and permit packet size adjustment for adaptation to varying error conditions.

In addition, NETBLT code must be modified to provide a standard stream-socket API with in sequence packet delivery and coordinate data transfer between applications and protocol. Reliable multicasting is also necessary but is left for future development (Section 5).

The remainder of this section addresses the modifications selected for small and large message operation respectively.

Small Message Operations

Connection and termination overhead is reduced for small message operation by providing a transaction-oriented connection/termination service. The service accommodates one or more packets up to a buffer of packets. As with large messages, the small message service must guarantee data delivery from the perspective of both receiver and sender, allow an active opener to send data, and provide reliable internet operation. An active opener is also allowed to request and receive data but this case is less important and is not addressed here.

The performance benefit of the service is reduction in the minimum number of data and control packets needed to transfer one data packet from 7+ for standard, large-message operation to 3 with opener sending. The service also allows packet sizes to be reduced and reduces the

transaction delay needed to transfer one data-packet from 2 to 1 RTT for a sender and 1 to no RTT for a receiver.⁷

The small message transfer service operates by using a connection counter in an opening data packet in a similar manor to T/TCP [4,5]. A connection counter (`cc_count`) is incremented and sent with each new connection in either a `CC` or `CCnew` parameter in the packet header. The choice of `CC` or `CCnew` parameter is determined based on `cc_sent`(to a passive opener, PO) and `cc_rcv`(from an active opener, AO) variables containing prior sent and received `cc_counts`.

`CCnew` is sent when the AO has no `cc_sent` and `cc_rcv` values for the PO or when its `cc_count` value is less than the last `cc_count` sent to the PO (because of number rapping). The variable `cc_sent` for the PO is updated with `cc_count` only if the PO accepts the AO's `cc_count` in its `cc_rcv` variable for the AO. `cc_rcv` is updated with a received `cc_count` only if it is: 1) sent in a `CCnew` parameter or 2) sent in a `CC` parameter that satisfies $CC < cc_rcv(AO)$.

A PO will accept a connection attempt immediately if $CC > cc_rcv(AO)$. Otherwise, the AO's data is stored and the connection will be accepted if the PO receives an echo of its `cc_count` subsequently. In its responses to the AO, the PO will send its `cc_count` in its `CC` variable, echo the AO's `cc_count` in a `CCecho` variable, and provide an indication of its `CC` test outcome.

Figure 2 illustrates the functional flow for a successful transaction-oriented connection. The capitalized variables in the figure represent specific packets needed for small message transfer. A description of the packet types is beyond the scope of this paper.

Figure 2 shows how a reliable connection can be established immediately without first conducting a standard three-way handshake. Figure 3 shows a state diagram for the active open send operation and indicates the same connection functionality addressed in Figure 2.⁸

⁷ The header size needed for data packet reduces to only 16 bytes.

⁸ Note the following additional functions. A normal connection is needed when connection parameters require updating. Also, for internet operation, a transaction connection will typically be short enough to avoid requiring a long connection completion wait to mitigate false acceptance of old transaction-open packets (as in T/TCP).

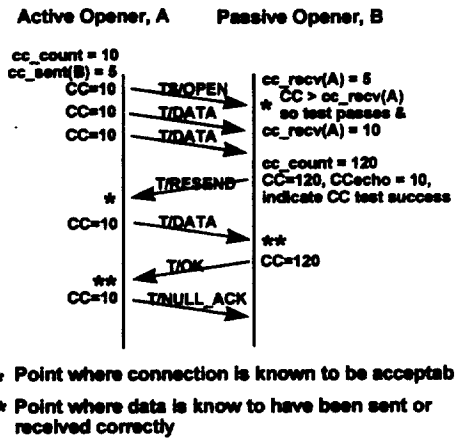


Figure 2. Functional-Flow for a Successful Transaction-Oriented Connection

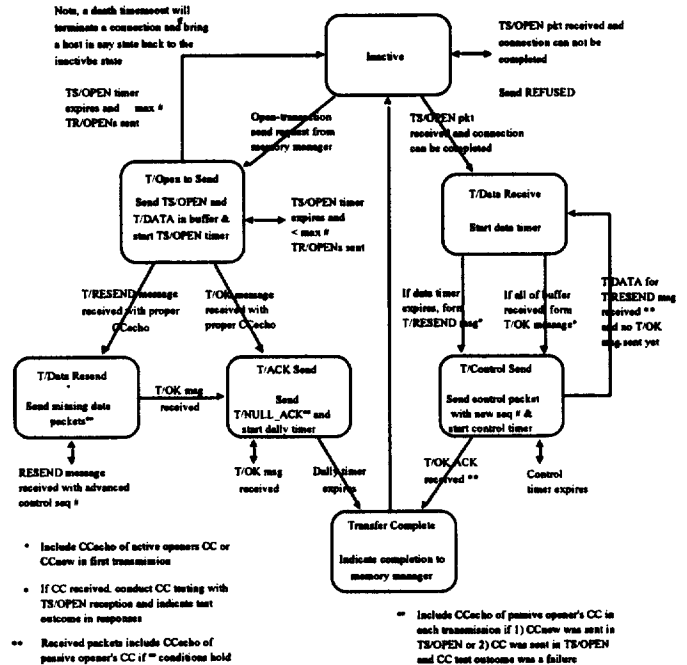


Figure 3. Small Message Active-Open-Send Operation

Large Message Operations

Standard, large-message operation is made robust for internet operation by providing three-way handshaking for normal connection establishment and by resolving simultaneous connection and termination requests.

Three-way handshaking is provided by using a transaction-oriented `CC` variable in combination with a `CCecho` as shown in Figure 4. The capitalized variables in the figure represent large-message packets; there

description is beyond the scope of this paper. Note, however, CC and CCecho variables are included in packet headers and the CC variable replaces the standard NETBLT connection ID. In addition a packet size variable is included to allow packet size adaptation and a selective packet error mask is optionally included (for large messages with 32 or fewer packets) to reduce overhead.

Current development efforts are focused on modifying TACO2 NETBLT code to include a standard stream socket API. Future efforts will include modifying the code to include the modifications described in the paper and determining the best adaptive algorithms and parameter settings for supporting efficient, responsive operation in tactical conditions. Protocol timer estimation, rate control adaptation, and packet size adaptation will be investigated. Support for reliable multicasting will follow.

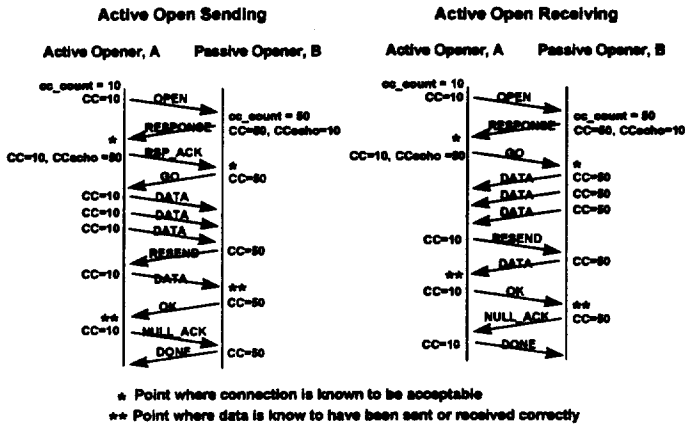


Figure 4. Functional-Flow for Standard Connections with Robustness Improvements

The state diagram for standard, large-message operation with internet robustness features is shown in Figure 5. The state diagrams for receiver and sender data transfer are not shown because of space constraints.

Resolution of simultaneous connection and termination requests are handled by using CC variables to force a host with a smaller CC value to take a passive role in the connection or termination.⁹

SUMMARY

This paper presents the information about initial development of a reliable, efficient data transport protocol for tactical communications systems. The information includes requirements, protocol comparisons, and needed NETBLT/baseline modifications. The modifications are focused on efficient transfer of small messages using a transaction-oriented approach and robust large message operation using a standard NETBLT connection approach.

⁹ A sudden, non-graceful termination can also occur by the transport protocol or its application. The transport protocol causes immediate termination as the result of an error or death time-out.

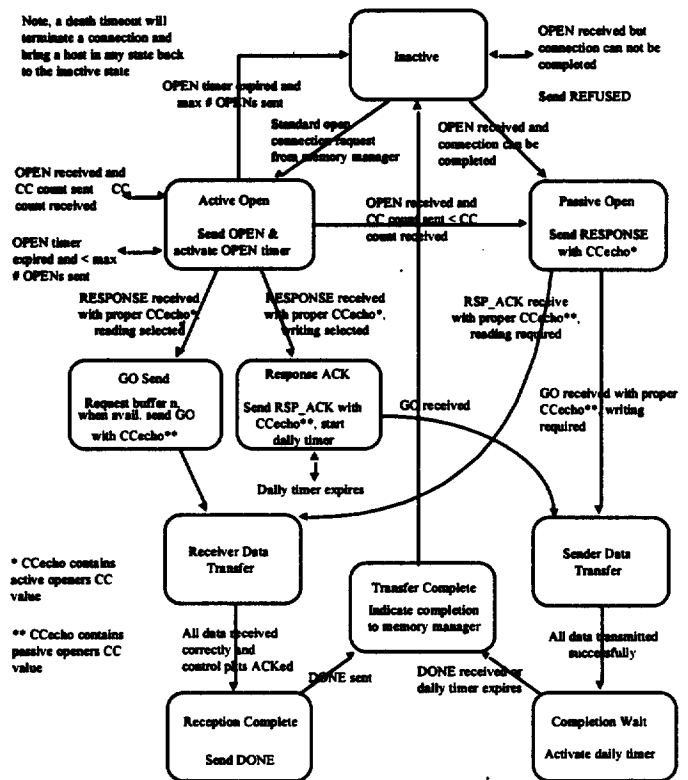


Figure 5. Standard, Large-Message Operation

REFERENCES

- Clark, D. D., Lambert, M. L., Zhang, L., *NETBLT: A Bulk Data Transfer Protocol*, RFC 998, March 1987.
- Tactical Communications Protocol 2 (TACO2)*, MIL-STD-2045-44500, 18 June 1993.
- Cheriton D., *VMTP: Versatile Message Transaction Protocol*, RFC 1045, February 1988.
- Stevens, R. W., *TCP/IP Illustrated, Volume 3*, Addison-Wesley, New York, 1996
- Bradon R., *T/TCP - TCP Extensions for Transactions Functional Specification*, RFC 1644, July 1994.