Chapter 5

TRUMP

This chapter outlines the structure of TRUMP[1] a new message-based transport protocol designed by the author to meet the requirements for a Transport Layer given in the previous chapter. As well, TRUMP has been developed to be a fully-fledged protocol in its own right. TRUMP has other features not required by the framework. A partial verification of the protocol is described in Appendix B, and a possible set of headers to use TRUMP within an IPv6 network is given in Appendix C.

TRUMP's features are:

- Error detection and recovery using selective acknowledgment and retransmission.
- Explicit and implicit connection setup and connection teardown.
- Rate-based flow control, using information passed to TRUMP by the Network Layer.
- An infinite amount of in-transit data: the source can transmit all of a message before any of the acknowledgments have arrived. This aids high throughput.
- Use of protocol timers which are not based on the round-trip time.
- Up to 16 possible types of transport header/data encryption. The encryption covers all fields except for the protocol version number, the encryption type and the segment checksum.

TRUMP is a lightweight transport protocol that transports messages from one communication endpoint (the *source*) to another (the *destination*). This simple unidirectional message passing[2] allows higher layers to provide bidirectional message passing, remote procedure calls and streams (i.e virtual circuits).

TRUMP does not need an explicit call setup; this is to avoid the overhead of call setup in Distributed Systems where the message size is often small. Instead, call setup can be implicit: here every datagram sent from the source to the destination contains data which is part of the message. Thus, the minimum packet exchange to pass a message is 2 packets: one from the source containing the data, and the other from the destination containing an acknowledgment. Each packet is acknowledged: however, acknowledgment packets may acknowledge more than one packet.

Explicit connection setup and teardown is also provided by TRUMP. This allows negotiation of connection privileges and initial flow control values.

[1] a reshuffled acronym standing for **R**eliable **U**nicast **T**ransport **P**rotocol.

[2] Protocols such as TCP and TP4 associate two data streams between a pair of connected endpoints; one in each direction.

Because TRUMP transmits unidirectionally, the following sections discuss the protocol from the point of view of a *source* sending a message to a *destination*.

5.1 Features of TRUMP

5.1.1 Implicit and Explicit Connection Setup and Teardown

TRUMP allows both implicit and explicit connection setup and teardown. For small messages (such as client-server interactions), implicit connection setup is employed: the source begins transmission of the data packets containing the message; the destination creates the data structures needed to reassemble and deliver the message after reception of the first data packet. If the destination cannot accept the message for any reason, it returns a special acknowledgment packet indicating failure of the implicit connection. Otherwise, it begins to return acknowledgments of successful packet reception.

With explicit connection setup, the source initiates the connection by transmitting a connection setup packet to the destination. The destination returns an acknowledgment to this packet permitting or denying the connection. The destination's acknowledgment should also return initial flow control data that the source can use for data transmission. When a successful connection acknowledgment arrives, the source may commence transmission of the message.

Implicit connection teardown is available in TRUMP: when the last acknowledgment for a message arrives at the source, both the source and destination assume that the connection is over. The destination implementation may retain the message's data structures for a period in order to catch any duplicate or delayed packets from the source.

Explicit connection teardown can be selected by the source, or by the destination if explicit connection setup was used. If chosen by either side, the source must acknowledge the last acknowledgment for a message from the destination; this informs the destination that the connection has been terminated by the source.

5.1.2 Connection Identifiers

During connection setup, TRUMP negotiates a connection number for the connection, and a segment sequence number for the first packet. The connection number uniquely identifies the connection[3] between the source and destination. While not currently used, future versions of TRUMP may use this field as an aid to perform CSLIP-style header compression [Jacobson 90].

Messages passed to TRUMP are segmented, with each segment forming the payload of a TRUMP packet. The segments are numbered incrementally, with the first segment sequence number chosen randomly by the source. With a sequence space of 32-bits, this allows a message to be composed of over 4 billion segments.

5.1.3 Selective Retransmission

Error detection and recovery is performed using *selective retransmission*. TRUMP messages are dynamically broken into *segments*: each segment is transmitted in a TRUMP packet. The last segment of a message is marked as such.

[3] i.e uniquely identifies the connection for both the source and destination.

The destination can acknowledge up to 16 segments, indicating which segments have been lost. Lost segments are retransmitted by the source, along with previous segments whose acknowledgments have not been received from the destination. The latter is used to sustain throughput, as sending a query packet to 'obtain' lost acknowledgments would delay the source by one round trip time.

5.1.4 Qualities of Service

TRUMP provides two qualities of service: 'time-sensitive' communications and reliable communications. With both qualities, a TRUMP destination reassembles the message from its constituent segments, reordering segments and discarding duplicates as necessary.

A TRUMP source retransmits lost segments only for messages being sent in the reliable mode. This implies that for time-sensitive communications, the destination may receive a correctly-ordered message with sections missing. Here the destination can pass the segments received 'as is' to upper communication layers.

The characteristics for each service quality are shown in the following table.

Service Quality	Destination ACKs	Source Retransmits	Incomplete Messages
Time-Sensitive	Yes	No	Yes
Reliable	Yes	Yes	No

5.1.5 Dynamic Fragmentation and MTU

Messages are not totally segmented before transmission, as the Maximum Transmission Unit of the underlying network between the source and destination may vary during the message transmission. Instead the transport protocol keeps a current *MTU* value for the underlying network between the source and destination, and uses this to dynamically segment the message. This MTU can be updated from information passed on by the Network Layer.

The MTU may decrease during transmission. In this case, several transmitted segments may not be received because the exceeded the new MTU. TRUMP is able to break a segment into 2 to 16 *fragments*, and retransmit them as new segments. TRUMP's headers include enough information to allow the original segment to be reassembled from the fragments.

5.1.6 Flow Control and Throughput

TRUMP employs rate-based flow and control to maintain its rate at a value which can be sustained by both the destination and the underlying network. Flow and congestion control is also separated from error detection and recovery. Data is transmitted continuously, rather than stopping transmission to wait for acknowledgments which will take one round trip time to arrive, and will slow down throughput.

Flow and congestion control is performed by a variation of Leaky Bucket [Turner 86] which has unlimited capacity (i.e, TRUMP does not discard data from the application layer): the inter-packet delay is set from the congestion information supplied by the network according to Equation 4.1. The Leaky Bucket ensures that the bit rate of the transmission is even.

If the source increases its inter-packet delay, it ignores network congestion information in acknowledgments for packets transmitted before the delay change. This allows one round-trip for the effect of the delay change to be measured by the network.

Flow and congestion control is performed on a source/destination basis, and a source may use the transmission delay interval for packets to one destination to transmit packets to other destinations, if the delay is great enough. The Leaky Bucket flow control mechanism would typically be implemented by a 'callout' system [Vahalia 96], where one timer controls the transmission of the next packet in a queue of packets ordered by desired transmission time.

In the situation where explicit call setup is not performed, the determination of the initial inter-packet interval is left unspecified.

5.1.7 Unacknowledged Data

Unlike traditional transport protocols such as TCP or TP4, there is **no** protocol-defined "window size", i.e. no limit on the amount of unacknowledged data at the source. Theoretically, all segments of a large message could be sent before any acknowledgment packets arrive from the destination. Any limit on the amount of unacknowledged data is therefore implementation-dependent.

5.1.8 Integration with Traditional and New Communication Systems

TRUMP has been designed to integrate with both traditional and distributed communication systems. In order to achieve this, TRUMP's headers allow communication endpoint identifiers to be 0, 16, 32 or 64 bits in size. A TRUMP implementation can choose to support one or more of these identifier sizes.

5.1.9 Security

TRUMP's headers provide for the encryption of both the data and the transport protocol headers themselves. Currently, two encryption systems are defined: no encryption, and per-segment DES block encryption. Fourteen other encryption options can be added in the future. A TRUMP implementation can choose to support one or more of the defined encryptions. Key negotiation and management are not covered by the TRUMP specification.

The size of the initial segment sequence number, and its pseudo-random choice by the source also makes connection setup spoofing very difficult.

5.2 Connection Setup

The following sections describe the operations performed as TRUMP goes through the stages of connection setup, data transmission and connection termination. State diagrams which outline the operations involved in peer-to-peer communication for each stage can be found in Section B.5.

TRUMP provides both implicit and explicit setup of data exchange connections. A connection is explicitly set up when a source sends a **SETUP** packet, which is answered by a **SACK** packet from the destination, as in Figure 7.

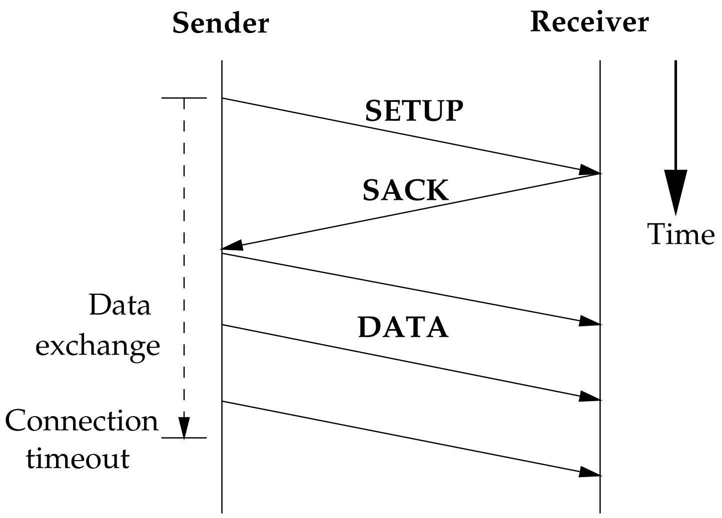

Figure 7: Explicit Connection Setup

The **SETUP** packet holds the connection number for the connection, the initial segment sequence number, the desired quality of service, and the identifiers of the transport-level source and destination communication endpoints. Explicit connection teardown can also be selected by the source or destination with fields in each of the **SETUP** and **SACK** packets.

The **SACK** packet indicates to the source if the connection has been successful, and if not, why not. If the connection is unsuccessful, the source must not transmit any further packets for the connection.

If connection setup is not chosen by the source, implicit connection begins with the transmission of data. Here, the first **DATA** packet is marked as being an implicit **SETUP** packet, and holds the fields mentioned above. When the **DATA** packet is not an implicit **SETUP** packet, the QOS field is invalid.

During connection setup, a destination can discard data packets for the connection before the implicit/explicit **SETUP** packet; this may happen when packets arrive out of sequence.

During implicit or explicit connection setup, the source may set a timer to indicate whether or not the destination is reachable. The value of the timer is implementation-dependent and not specified by the TRUMP protocol.

5.3 Data Transmission

Data transmission in TRUMP is characterised by a source sending a stream of packets to the destination, which returns acknowledgment packets back to the source for a number of data packets, as shown in Figure 8.

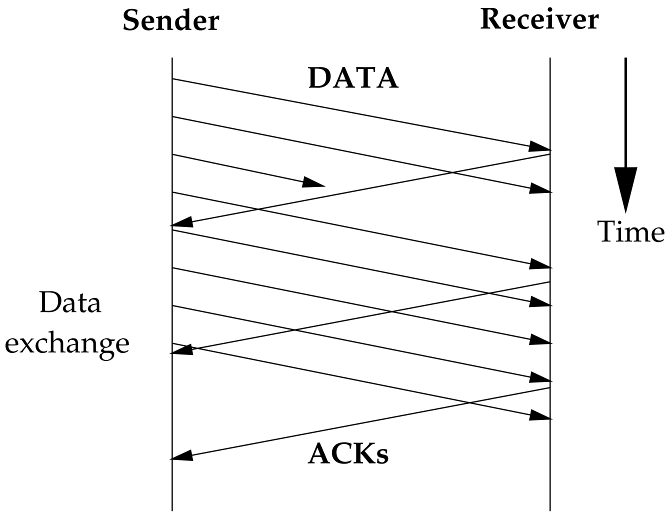

Figure 8: Data Transmission

The transmission of data can be divided into those operations performed by the destination, and those performed by the source.

5.3.1 Destination Operation

The destination is totally driven by packets received from other machines. It must check the checksum on received packets, acknowledge data segments once they arrive, reorder segments and reassemble the message from them.

Not including retransmission, segments are transmitted with monotonically increasing *segment sequence numbers*. Acknowledgment packets hold acknowledgment bitmaps of 1 to 16 bits; thus TRUMP can can acknowledge up to 16 segments in each ACK packet. If bit $G[x]$ is on, the segment with sequence number x was received successfully, otherwise it was not received.

Two sequence numbers are returned with the bitmap; B is the sequence number of the first bit in the bitmap, and T is the highest valid sequence number in the bitmap. Note that $B \leq T \leq B + 15$.

The TRUMP protocol allows the destination to return acknowledgment bitmaps of up to 16 bits, but this maximum is not always achievable. On a real network, a destination will get duplicate and out-of-order packets. The destination operates as follows:

- If a packet with sequence number S arrives such that $S < B$ or $S > B + 15$, then it cannot be acknowledged within the existing partially-filled acknowledgment bitmap. Therefore, the existing acknowledgment, with B, T and the bitmap G, is returned to the source immediately, and a new acknowledgment is created as follows.
- If a packet with sequence number S arrives, and there is no existing acknowledgment bitmap, one is created with $B \leftarrow S, T \leftarrow B, G[S] \leftarrow 1$ and all other bits in the bitmap turned off.
- If the above two conditions are false, $G[S] \leftarrow 1$, and $T \leftarrow S$ iff $S > T$.
- If $G[S]$ was already set on, an acknowledgment is returned immediately with B, T and the bitmap G.

Finally, the data segments are reassembled by the destination. According to the negotiated quality of service, fully or partially received messages are passed to higher protocol layers.

5.3.2 Source Operation

The source is driven by:

- Messages passed to it from higher layers,
- Acknowledgment packets from the destination,
- Sustainable rate information provided by the Network Layer, and
- Timeouts on unacknowledged data.

Messages passed from higher layers to a TRUMP source specify the source and destination communication endpoints (and network addresses), and the quality of service required. If the source has no initial sustainable rate or maximum transmission unit about the network to the destination, it may be able to ask the Network Layer to provide this information. This mechanism is left unspecified in the TRUMP protocol. Alternatively, an explicit connection may provide the source with this information.

The source then begins to segment the message. Each segment is transmitted, with the first octets of successive packets separated by the current inter-packet delay, to limit the throughput to that sustainable by the network and the destination.

Fragmentation and transmission continues until there is no new data from the higher-level application to send to the destination. Here, the source may choose to retransmit outstanding segments, in order to prompt the destination. These should be transmitted cyclically in their original transmission order, spaced by the current inter-packet delay for the connection.

Acknowledgment packets from the destination:

- Possibly update the inter-packet delay,
- Possibly update the maximum transmission unit, and
- Cause selective retransmissions to occur, if needed.

A TRUMP source keeps three linked lists of segments:

- The **lost** list holds the list of segments known to have been lost,
- The **not-lost** list holds the list of segments in transit, but not known to be lost, and
- The **not-sent** list holds the list of segments not yet sent by the source.

Segments marked as lost in an acknowledgment are moved to the tail of the lost list. The choice of which segment to transmit next is as follows:

1. If the lost list is not empty, choose the segment at its head, transmit it, and move it to the tail of the not-lost list.
2. If the lost list is empty and the not-sent list is not empty, choose the segment at the head of the not-sent list, transmit it, and move it to the tail of the not-lost list.
3. If both the lost and not-sent lists are empty, choose the segment at the head of the not-lost list, transmit it, and move it to the tail of the not-lost list.

4. If all three lists are empty, then no packets are available to transmit. If the application has requested that the connection be closed, TRUMP can perform connection teardown.

Note that when a time-sensitive quality of service is chosen, no segments are placed on the lost list, and it is always empty.

The third choice above causes TRUMP to lazily retransmit in-transit segments until they have been acknowledged by the destination. This mechanism prompts self-clocking acknowledgments from the destination, and avoids the use of a round-trip timer in the TRUMP source. Although this unnecessary retransmission of data goes against the design requirements of the rate-based framework, it does prove that no round-trip timer is required to perform error recovery.

Lazy retransmission can be replaced or augmented with a very coarse-grain round-trip timer, to prompt the source to retransmit some segments if the destination does not respond. Accurate round-trip estimation is not required.

5.4 Connection Teardown

The last packet of data from the source is marked as such, and indicates the end of data. Data exchange is considered to be complete if the destination has received all data segments, and the source has received acknowledgments for all data segments.

At the point when a TRUMP destination implementation has all portions of a message, it sets a close timer which is used to detect further packets for the connection: these might have been caused by a loss of acknowledgments from destination to source. The destination should acknowledge further packets from the source to complete the data exchange. Once the close timer expires, the destination assumes data exchange is complete, and can release its internal data structures as required. This is an implicit connection teardown. The value of the timer is implementation-dependent and not specified by the TRUMP protocol.

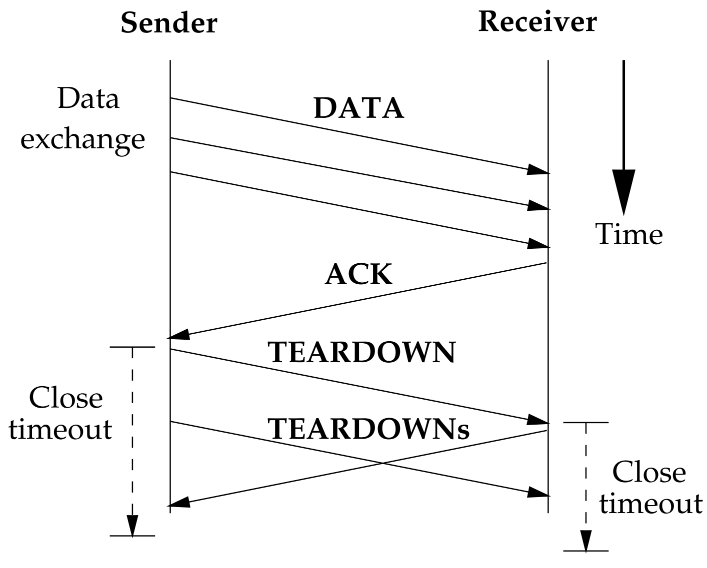

Figure 9: Explicit Connection Teardown

If explicit connection teardown was requested by either side and once the source has all acknowledgments, it sends a **TEARDOWN** packet to the destination. If/when this is received, the destination can formally close the connection as if the close timer had expired. It returns the **TEARDOWN** packet back to the source as well. This is shown in Figure 9.

The source is free to transmit **TEARDOWN** packets periodically, to ensure that the destination receives one. It must not exceed the value of the last **Return_Rate**. The destination must return all received **TEARDOWN** packets back to the source.

The source is also free to set an implementation-dependent close timer during **TEARDOWN** packet transmission. If the timer expires before a **TEARDOWN** packet is received from the destination, the source can assume that the connection is successfully closed.

5.5 Abnormal Connection Termination

At any stage during the exchange of data, a destination may abnormally terminate the exchange and close the connection. This is done by returning to the source a **SACK** packet indicating the reason for connection termination. All new packets from that connection will then be discarded by the destination. Upon receipt of the **SACK** packet, a TRUMP source discontinues the data exchange, marks the connection as closed if necessary, informs higher communications layers of the error, and discards any buffers and data structures used during the connection.

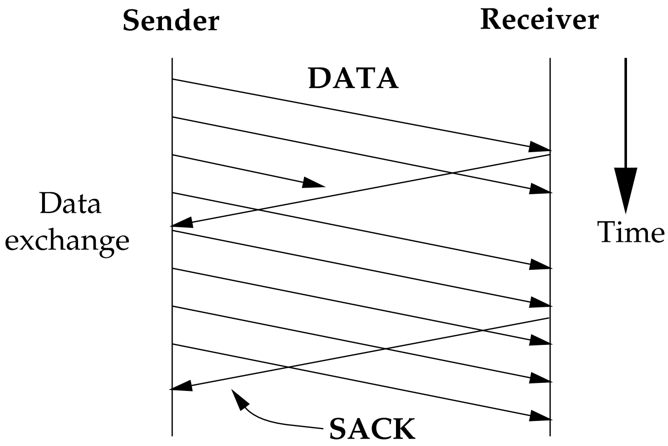

Figure 10: Abnormal Connection Termination

A destination implementation is permitted to retransmit the **SACK** packet several times if data packets for the connection continue to arrive. Source implementations must cope with the receipt of several **SACK** packets for the same connection.

5.5.1 End-to-End Flow Control

As described, TRUMP provides no end-to-end flow control. A sliding window flow control mechanism could be utilised, but this would be undesirable in view of the design requirements of the rate-based congestion control framework. A rate-based form of flow control would suit TRUMP better.

In every TRUMP header returned by the destination to the source, a **Flow_Rate** field holds the highest bit rate for which the destination can accept TRUMP packets from the source. The TRUMP source must use the most recent **Flow_Rate** field, and must not exceed this rate in the transmission of TRUMP packets to the destination.

The mechanism which sets the **Flow_Rate** field in the destination is not specified. The destination may use such techniques as Multiplicative Decrease/Additive Increase, or it may use techniques which estimate the rate of input buffer draining by applications. The source remains oblivious to the mechanism employed.

Note that a TRUMP source must obey both the **Flow_Rate** set by the destination, and the **Return_Rate** set by the Network Layer. In practice, the minimum of the two must be used. As well, a TRUMP source must not advertise a **Desired_Rate** higher than the most recent **Flow_Rate** value. This ensures that network capacity is not wasted when a flow is bottlenecked by its destination.

5.6 Summary

TRUMP, as a transport protocol, has been designed not only to provide the rate-based flow control as required in the proposed congestion control framework, but to provide other services and features useful to applications. This chapter has described most of TRUMP's features, including those not relevant to congestion control.

The next chapter resumes the exposition of the rate-based framework with a discussion on the central element of the framework, the Sustainable Rate Measurement function which exists in every network router.