Congestion Control in Linux TCP

plemented optionally.

Building up a single consistent protocol

of outstanding segments by sending a new segment for each incoming acknowledgement, if the congestion

gests that if an acknowledgement for a retransmitted segment echoes a timestamp earlier than the timestamp of the retransmission stored at the sender, the original segment has arrived at the receiver, and the retransmission was unnecessarily made. In such a case, the TCP sender can continue by sending new data and revert the recent changes made to the congestion control parameters.

Instead of inferring congestion from the lost packets, *Explicit Congestion Notification (ECN)* [RFB01] was suggested for routers to explicitly mark

reduce its transmission rate to mit-

congestion in the netw ork. ECN allo ws the

ers to be congestion-a w are without necessar -

g from Ex-

segment in the congestion window, regardless of its size. Therefore, Linux is more conservative compared to the byte-based approach when the TCP payload consists of small segments.

The Linux TCP sender uses the same set of concepts and functions for determining the number of outstanding packets with the NewReno recovery and with the two flavors of SACK recovery supported. When the SACK information can be used, the sender can either follow the Forward Acknowledgements (FACK) [MM96] approach considering the holes between the SACK blocks as lost segments, or a moreCK

is

SACK, which is not too aggressive in a network involving reordering.

Table 1: TCP congestion control related IETF specifications implemented in Linux. + = implemented, * = implemented, but details differ from specification.

Specification	**Status**
RFC 1323 (Perf. Extensions)	+
RFC 2018 (SACK)	+
RFC 2140 (Ctrl block sharing)	+
RFC 2581 (Congestion control)	*
RFC 2582 (NewReno)	*
RFC 2861 (Cwnd validation)	+
RFC 2883 (D-SACK)	+
RFC 2988 (RTO)	*
RFC 3042 (Lim. xmit)	+
RFC 3168 (ECN)	