Requirements for Scalable Transport	Normative text	Linux TCP-Prague	Evaluation/Remarks/Plans/Issues/Objections
1. Use of L4S Packet Identifier (A1.1)	Section 4.1: A sender that wishes a packet to receive L4S treatment as it is forwarded, MUST set the ECN field in the IP header (v4 or v6) to the ECT(1) codepoint.	Compliant Kernel updated to provide an option via the congestion control module (reusable for other CCs in Linux)	Implementation for setting ECT(1) works, no issues found. The only known "safety" issue is when a Classic-ECN-only bottleneck is active in the path.
2. Accurate ECN Feedback (A1.2)	Section 4.2: For a transport protocol to provide scalable congestion control it MUST provide feedback of the extent of CE marking on the forward path.	Compliant Kernel updated to support accurate ECN (reusable for other CCs in Linux)	Uses only the tcp-flags feedback. No option fields implemented.
(Scalable CC requirement)	Section 4.3: As a condition for a host to send packets with the L4S identifier (ECT(1)), it SHOULD implement a congestion control behaviour that ensures that, in steady state, the average time from one ECN congestion signal to the next (the 'recovery time') does not increase as flow rate scales, all other factors being equal.	Compliant Reuses DCTCP reduction scheme proportional to ECN marks that converges at 2 marked packets per RTT (when marking is sufficiently randomized)	Current Rtt-independent target limits the feedback to 2 marks per 25ms. Assuming minimal realistic RTTs on the internet (10ms?) it is still about one mark per RTT (in steady state). A lower target-RTT can be considered, and a faster additive increase implementation can improve search for capacity.
(ECT(1) use needs Prague compliance)	Section 4.3: In order to coexist safely with other Internet traffic, a scalable congestion control MUST NOT tag its packets with the ECT(1) codepoint unless it complies with the following bulleted requirements.	Compliant (not for one SHOULD) See below compliance statements of items 3 to 7 and "Burst limit"	The minimum window "SHOULD" requirement is less essential when RTT-independence with pacing is implemented, and AQMs apply drop when marking probabilities become saturated.
(Prague compliance description)	Section 4.3: The specification of a particular scalable congestion control MUST describe in detail how it satisfies each requirement, and for any non-mandatory requirements, it MUST justify why it does not comply.	Non-compliant ICCRG Prague-CC in progress. This document provides a high-level description of the Prague functionality and public code is available for details.	Not an implementation related requirement. Is this requirement really needed?
3. Fall back to Reno-friendly congestion control on packet loss (A1.3)	Section 4.3: As well as responding to ECN markings, a scalable congestion control MUST react to packet loss in a way that will coexist safely with a TCP Reno congestion control [RFC5681].	Compliant Implemented in Prague and also been backported to DCTCP for all active supported Linux kernels	No issues.
4. Fall back to Reno-friendly congestion control on classic ECN bottlenecks (A1.4)	Section 4.3: A scalable congestion control MUST implement monitoring in order to detect a likely non-L4S but ECN-capable AQM at the bottleneck. On detection of a likely ECN-capable bottleneck it SHOULD be capable (dependent on configuration) of automatically adapting its congestion response to coexist with TCP Reno congestion controls [RFC5681]. To participate in the L4S experiment, a scalable congestion control MUST be capable of being replaced by a Classic congestion control (by application and by administrative control).	Compliant A monitoring scheme has been implemented based on RTT fluctuation heuristics and is available when an option is enabled. A fallback mechanism triggered by the monitoring has been implemented and is available when an option is enabled. Heuristics parameter options are available for further tuning and customization for network conditions.	Too many false hits. Needs real deployment experience to understand the real extend of this issue, and to tune/customize the settings based on dedicated experienced problems. Safety issues should be handled more on an operational level (A/B testing, active probing, network monitoring, …).
5. Reduce RTT dependence (A1.5)	Section 4.3: A scalable congestion control MUST eliminate RTT bias as much as possible in the range between the minimum likely RTT and typical RTTs expected in the intended deployment scenario.	Compliant Implemented in Prague and further tunable with options and extensible with new pluggable RTT conversion functions (3 example functions are implemented). Current default is making all flows with an RTT lower than 25ms behave as a flow with an RTT of 25ms after 500 RTTs. This results in convergence to an equal rate for all Prague flows that have an RTT lower than 25ms and all Classic flows with exactly 25ms RTT.	Works perfect for RTTs that are below the target RTT. Beyond requirement: When the RTT of a Prague flow is a tenfold more than the target RTT, the DCTCP scheme penalizes throughput more for those high RTT flows and only when the low RTT flows get synchronized due to the STEP (bursts of non-randomized marks). This can be solved, but do we want/need very high RTT flows making use of a low latency service?? This claimed issue is currently not a requirement.

6. Scaling down to fractional congestion windows (A1.6)	Section 4.3: A scalable congestion control SHOULD remain responsive to congestion when typical RTTs over the public Internet are significantly smaller because they are no longer inflated by queuing delay.	Compliant code exists but not used A research implementation has been done, but has not been integrated, as it disadvantages Prague compared to Classic flows that preserve a minimum window of 2 packets, when mixed with non-responsive traffic. RTT-independent CC-behavior and the revert-to-drop-on-high-congestion AQM behavior eliminate largely the need for this requirement.	No issues known when with the current implementation that has RTT-independent code and when used with a reverting to drop AQM. Consider making this Prague Requirement a Performance Optimization to avoid drop when the RTTs and queues are very small for specific deployments, and the minimum window is expected to frequently drive an AQM into drop mode.
7. Measuring Reordering Tolerance in Time Units (A1.7)	Section 4.3: A scalable congestion control SHOULD detect loss by counting in time-based units, which is scalable, as opposed to counting in units of packets (as in the 3 DupACK rule of RFC 5681 TCP), which is not scalable. This requirement does not apply to congestion controls that are solely used in controlled environments where the network introduces hardly any reordering.	Compliant RACK is default enabled in Linux	No issues are identified for Prague using RACK.
(Burst limit)	Section 4.3: A scalable congestion control is expected to limit the queue caused by bursts of packets. It is only required that the specification of a particular scalable congestion control MUST define, quantify and justify its approach to limiting bursts.	Compliant Pacing is default enabled in Prague and TSO burst sizing is adapted to cause maximum 250us delay at the current rate (assumed bottleneck serving rate)	Bursts from the sender need to be controlled to support the low thresholds. The current default DCTCP implementation in Linux causes a lot of bursts and achieves consequently low link utilization. If networks are causing bursts, the thresholds on the AQMs need to be adjusted, and future network technology needs to take the new L4S congestion control capabilities into account (that it can support very low queue thresholds if the network can avoid that level of bursts).
Scalable Transport Protocol Optimizations	**Appendix text (no normative refs)**		
1. Setting ECT in TCP Control Packets and Retransmissions (A2.1)	This item only concerns TCP and its derivatives (e.g. SCTP), because the original specification of ECN for TCP precluded the use of ECN on control packets and retransmissions. To improve performance, scalable transport protocols ought to enable ECN at the IP layer in TCP control packets (SYN, SYN-ACK, pure ACKs, etc.) and in retransmitted packets. The same is true for derivatives of TCP, e.g. SCTP.	Compliant Prague enables ECT(1) on all packets	No issues are identified for Prague using ECN++.
2. Faster than Additive Increase (A2.2)	It would improve performance if scalable congestion controls did not limit their congestion window increase to the standard additive increase of 1 SMSS per round trip [RFC5681] during congestion avoidance. The same is true for derivatives of TCP congestion control, including similar approaches used for real-time media.	Research code exists	The current public Linux TCP Prague version (not including any further research code) can benefit from faster increases. While not a safety requirement, the current performance is often claimed (indirectly) as an “issue” on the list.
3. Faster Convergence at Flow Start (A2.3)	Particularly when a flow starts, scalable congestion controls need to converge (reach their steady-state share of the capacity) at least as fast as Classic congestion controls and preferably faster. This affects the flow start behaviour of any L4S congestion control derived from a Classic transport that uses TCP slow start, including those for real-time media.	Research code exists	The current public Linux TCP Prague version (not including any further research code) can benefit from a better/gradual slow start exit strategy. While not a safety requirement, the current performance is often claimed (indirectly) as an “issue” on the list.