Ans 1.

Paper name/reference	Work done	Drawback
CLTCP: An adaptive TCP congestion control algorithm based on congestion level	Gave a new congestion control algorithm using TCP Reno, AIMD for high-BDP and lossy networks.	They didn't care about energy efficiency and load balancing (some paths are congested some are very light, so light lead to high energy loss).
Modified TCP Congestion Control Algorithm for Throughput Enhancement in Wired-cum-Wireless Networks(TCP Reno)	Determine optimal congestion window for TCP sender, given a function, which will determine the current window size during the window recalculation phase for an optimal fair share of bandwidth	Doesn't give a good performance on multiple packet loss. Too slow in start and drastic decrease on any packet loss.
TCP Vegas: End to end congestion avoidance on a global Internet	Provide an algorithm where they are sending the only ⅕ to ½ bytes retransmitted as compared in TCP reno	Unfair share of bandwidth, rerouting may increase propagation delay, ignore receiver data rate and keeps constant data rate
FAST TCP: Motivation, Architecture, Algorithms, Performance.	Maintain packet queue, if too few packets are there, increase the data rate, decrease data rate if too many packets are in queue	Sometimes show unfairness due to inaccuracy in propagation delay. Face frequent buffer overflow during less size of the buffer.

A thread synchronization model for the PREEMPT_RT Linux kernel (Motivation)

This paper proposed an automata-based model by which we can analyse, events, interrupt handling, preemption time, scheduling, locking etc and calculate the latency.

Another thing this paper proposed is the mechanism of kernel tracing, what sequence of events leads to any specific state. This model captures the events from the real-time execution of an operating system.

By testing and applying the above models they found 3 bugs in Linux kernel code which are:- 1. Inefficiency in scheduler, inaccurate old tracer (sometimes miss few events) and usage of error in mutex.

And last but not least they propose how a model can understand Linux properties in a pseudo way without analysing the whole code. The last thing is very essential Linux is not easy to understand, many time developers can't even understand the error so tracing will be a necessary role and always learning in a logical way is far easy to understand instead of whole code.

Shared Bottleneck-Based Congestion Control and Packet Scheduling for Multipath TCP(Drawbacks)

- Doesn't provide fairness in bandwidth sharing.
- Throughput maximization is possible when all are sharing the same bottleneck.
- OFO (Out of Order problem) if higher-order received and lower order received very late then our buffer will have to store all the higher-order for a very long duration. Different paths have different data rate so OFO is common.

- Since more paths will require more buffer size and more computation power at the receiver end, for example, IoT devices don't have this much computational power.

Scaling Open Source Communities: An Empirical Study of the Linux Kernel (Suggestions)

- Decentralization of community (different groups for different feature code review for example 1 team for the editor, 1team for compilers).
- We can use blockchain way where 51% nodes approve the transaction that it shows verified, we can also do something like this, where 51% trusted developer or organization approve the code we can add it directly to the main branch.
- A huge set of test cases should be provided and ask contributors to send the result file of that test case and then maintainers should verify the pull request.