Store-and-forward waits to receive whole packet before starting to forward it, and so can check for errors and prevent errored packets being forwarded. Cut-through has lower delay: only waits to start forwarding it as soon as outgoing port is determined by classifying packet from header. Cut-through forwards errored packets. Adaptive forwarding switches between cut-through and store-and-forward according to packet error rate.

Buffer memories need to be multi-access (for receiving and sending packets), high speed (to match line rate) and large & cheap. Video RAMs provide high-speed multi-port (mainly sequential) access to large amount of low-power & high-density DRAM. Loss of random access is OK for networking where read bytes of packet in sequence.

FIFOs may reduce throughput by packet at head-of-line blocking packets that arrived later but which are destined to idle output ports.

Discard packets when buffer is full, but need to decide which.
drop head of queue (oldest) for real-time UDP
drop tail of queue (newest) for TCP => retransmit less when go-back-N
drop packets marked by source as low priority (relative to others from that source)
drop all fragments of one datagram: badly damage 1 datagram rather than “slightly” damage many

Switch should limit buffer fill to contain delay and avoid filling it => large loss rate => tell source about congestion by dropping packets or ECN

Ends often interpret loss as a sign of congestion and respond by slowing down (e.g. TCP) => switch might drop packets solely to signal to source even when buffer space exists.
Active Queue Management / RED: Discard probability proportional to average queue length
proportional => higher when more congested; penalise sources that didn’t slow earlier
average => admit small bursts needed to get Slow Start going

Explicit Congestion Notification:
2 IP header bits carry info towards receiver: 4 states indicating if Congestion Experienced, incapable of ECN, 2 ECN capable states
2 TCP bits let receiver tell source CE, source tell receiver Congestion Window Reduced & determine if receiver supports ECN
Process:
1. Ends negotiate use of ECN (TCP SYN bits set & receiver replies only if supported).
2. Sender sends packets indicating ECN capable; alternates between 2 ECN capable states to send pseudo-random string
3. Congested switch replaces ECN-capable with CE
4. Receiver echoes IP CE status in TCP bit (e.g. in ACK). If CE, repeat until sender indicates CWR
5. Sender checks CE status from receiver. Receiver echoing ECN capable pseudo-random string => no congestion. Receiver can’t lie since can’t predict pseudo-random string. If receiver says CE, slow down & set CWR. Slow again if CE persists AFTER receive ACK for CWR packet.