



- [Help](#)
- [About V](#)
- [Commu](#)
- [Recent](#)
- [Contact](#)

Print/export

- Create a book
- Download as PDF
- Printable version

 Edit links

Article | [Talk](#)

[Read](#) [Edit](#) [View history](#)

Search 

From Wikipedia, the free encyclopedia

**This article has multiple issues.** Please help [improve it](#) or discuss these issues on the [talk page](#).

[hide]

- In [data networking](#) and [queueing theory](#), **network congestion** occurs when a link or node is carrying so much data that its [quality of service](#) deteriorates. Typical effects include [queueing delay](#), [packet loss](#) or the [blocking](#) of new connections. A consequence of the latter two effects is that an incremental increase in [offered load](#) leads either only to a small increase in network [throughput](#), or to an actual reduction in network throughput.<sup>[1]</sup>

Modern networks use [congestion control](#) and [congestion avoidance](#) techniques to try to avoid congestion collapse. These include: [exponential backoff](#) in protocols such as [802.11 CSMA/CA](#) and the original [Ethernet](#), [window reduction](#) in [TCP](#), and [fair queueing](#) in devices such as [routers](#). Another method to avoid the negative effects of network congestion is implementing priority schemes, so that some packets are transmitted with higher priority than others. Priority schemes do not solve network congestion by themselves, but they help to alleviate the effects of congestion for some services. An example of this is [802.1p](#). A third method to avoid network congestion is the explicit allocation of network resources to specific flows. One example of this is the use of Contention-Free Transmission Opportunities (CFTXOPs) in the [ITU-T G.hn](#) standard, which provides high-speed (up to 1 Gbit/s) [local area networking](#) over existing home wires (power lines, phone lines and coaxial cables).

[RFC 2914](#)  addresses the subject of congestion control in detail.

- 1 Network capacity
- 2 Congestive collapse
- 3 Congestion control
  - 3.1 Theory of congestion control
  - 3.2 Classification of congestion control algorithms
- 4 Mitigation
  - 4.1 Practical network congestion avoidance
  - 4.2 TCP/IP congestion avoidance
  - 4.3 Active queue management (AQM)
    - 4.3.1 Random early detection
    - 4.3.2 Robust random early detection (FRED)
    - 4.3.3 Flowbased-RED/WRED
    - 4.3.4 Explicit Congestion Notification (ECN)
    - 4.3.5 Cisco AQM: Dynamic buffer limiting (DBL)
    - 4.3.6 TCP window shaping
    - 4.3.7 Backward ECN (BECN)
  - 4.4 Side effects of congestive collapse avoidance
    - 4.4.1 Radio links
    - 4.4.2 Short-lived connections
- 5 See also
- 6 References
- 7 Books
- 8 External links

A fundamental problem is that all network resources are limited, including router processing time and link throughput.

- A **wireless LAN** is easily filled by a single **personal computer**
- Even on fast **computer networks** (e.g. **Gigabit Ethernet**), the backbone can easily be congested by a small number of servers and client PCs
- The aggregate transmission from **P2P** networks have no problem filling an **uplink** or some other network bottleneck
- **Denial-of-service attacks** by **botnets** are capable of filling even the largest **Internet backbone** network links, generating large-scale network congestion
- In **telephone** networks (particularly **mobile phones**), a **mass call event** can overwhelm digital **telephone circuits**

**Congestive collapse** (or **congestion collapse**) is a condition that a [packet-switched computer network](#) can reach, when little or no useful communication is happening due to congestion.

When a network is in such a condition, it has settled (under overload) into a stable state where traffic demand is high but little useful throughput is available, and there are high levels of **packet delay** and loss (caused by routers discarding packets because their output **queues** are too full) and general **quality of service** is extremely poor.

When more packets were sent than could be handled by intermediate routers, the intermediate routers discarded many packets, expecting the end points of the network to retransmit the information. However, early TCP implementations had very bad retransmission behaviour. When this packet loss occurred, the end points sent extra packets that repeated the information lost, doubling the data rate sent, exactly the opposite of what should be done during congestion. This pushed the entire network into a 'congestion collapse' where most packets were lost and the resulting throughput was negligible.

**Congestion control** concerns controlling traffic entry into a **telecommunications network**, so as to avoid **congestive collapse** by attempting to avoid oversubscription of any of the processing or **link** capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending **packets**. It should not be confused with **flow control**, which prevents the sender from overwhelming the receiver.

This section **does not cite** any **references or sources**. Please help improve this section by [adding citations to reliable sources](#). Unsourced material may be challenged and [removed](#). *(May 2013)*

Examples of "optimal" rate allocation are [max-min fair allocation](#) and Kelly's suggestion of [proportional fair allocation](#), although many others are possible.

The mathematical expression for optimal rate allocation is as follows. Let  $x_i$  be the rate of flow  $i$ ,  $C_l$  be the capacity of link  $l$ , and  $r_{li}$  be 1 if flow  $i$  uses link  $l$  and 0 otherwise. Let  $x$ ,  $c$ , and  $R$  be the corresponding vectors and matrix. Let  $U(x)$  be an increasing, strictly **concave** function, called the **utility**, which measures how much benefit a user obtains by transmitting at rate  $x$ . The optimal rate allocation then satisfies

$$\max_x \sum_i U(x_i)$$

such that  $Rx \leq c$

The **Lagrange dual** of this problem decouples, so that each flow sets its own rate, based only on a "price" signalled by the network. Each link capacity imposes a constraint, which gives rise to a Lagrange multiplier,  $p_l$ . The sum of these **Lagrange multipliers**,  $y_i = \sum_l p_l r_{li}$ , is the price to which the flow responds.

Congestion control then becomes a distributed optimisation algorithm for solving the above problem. Many current congestion control algorithms can be modelled in this framework, with  $p_l$  being either the loss probability or the queueing delay at link  $l$ .

A major weakness of this model is that it assumes all flows observe the same price, while sliding window flow control causes "burstiness" which causes different flows to observe different loss or delay at a given link.

## Classification of congestion control algorithms [edit]

*Main article: [Taxonomy of congestion control](#)*

There are many ways to classify congestion control algorithms:

- By the type and amount of feedback received from the network: Loss; delay; single-bit or multi-bit explicit signals
- By incremental deployability on the current Internet: Only sender needs modification; sender and receiver need modification; only router needs modification; sender, receiver and routers need modification.
- By the aspect of performance it aims to improve: high bandwidth-delay product networks; lossy links; fairness; advantage to short flows; variable-rate links
- By the fairness criterion it uses: max-min, proportional, "minimum potential delay"

## Mitigation [edit]

A couple of mechanisms have been invented to prevent network congestion or to deal with a network collapse:

- Network scheduler** – a program in **routers** to do **Active Queue Management** (that is, the arbitrary reorder or drop of network packets under overload)
- Explicit Congestion Notification** – an extension to the IP and TCP communications protocol, which adds a flow control mechanism upon which both ends react appropriately
- TCP congestion-avoidance algorithm** – several implementations of efforts to deal with network congestion

The correct end point behavior is usually still to repeat dropped information, but progressively slow the rate that information is repeated. Provided all end points do this, the congestion lifts and good use of the network occurs, and the end points all get a fair share of the available bandwidth. Other strategies such as **slow-start** ensure that new connections don't overwhelm the router before the congestion detection can kick in.

The most common router mechanisms used to prevent congestive collapses are **fair queueing** and other **scheduling algorithms**, and **random early detection**, or RED, where packets are randomly dropped proactively triggering the end points to slow transmission before congestion collapse actually occurs. Fair queueing is most useful in routers at choke points with a small number of connections passing through them. Larger routers must rely on RED.

Some end-to-end protocols are better behaved under congested conditions than others. **TCP** is perhaps the best behaved. The first TCP implementations to handle congestion well were developed in 1984,<sup>[*citation needed*]</sup> but it was not until **Van Jacobson**'s inclusion of an open source solution in the Berkeley Standard Distribution UNIX ("**BSD**") in 1988 that good TCP implementations became widespread.

**UDP** does not, in itself, have any congestion control mechanism. Protocols built atop UDP must handle congestion in their own way. Protocols atop UDP which transmit at a fixed rate, independent of congestion, can be troublesome. Real-time streaming protocols, including many **Voice over IP** protocols, have this property. Thus, special measures, such as **quality-of-service** routing, must be taken to keep packets from being dropped from streams.

In general, congestion in pure datagram networks must be kept out at the periphery of the network, where the mechanisms described above can handle it. Congestion in the **Internet backbone** is very difficult to deal with. Fortunately, cheap **fiber-optic** lines have reduced costs in the Internet backbone. The backbone can thus be provisioned with enough bandwidth to keep congestion at the periphery.<sup>[*citation needed*]</sup>

## Practical network congestion avoidance [edit]

Implementations of connection-oriented **protocols**, such as the widely used **TCP** protocol, generally watch for packet errors, losses, or delays (see **Quality of Service**) in order to adjust the transmit speed. There are many different network congestion avoidance processes, since there are a number of different trade-offs available.<sup>[?</sup>

### TCP/IP congestion avoidance [edit]

The **TCP congestion avoidance algorithm** is the primary basis for **congestion control** in the Internet.<sup>[3]</sup><sup>[4]</sup><sup>[5]</sup><sup>[6]</sup><sup>[7]</sup>

Problems occur when many concurrent TCP flows are experiencing **port queue buffer tail-drops**. Then TCP's automatic congestion avoidance is not enough. All flows that experience port queue buffer tail-drop will begin a TCP retrain at the same moment – this is called **TCP global synchronization**.

### Active queue management (AQM) [edit]

**Active queue management** (AQM) is the reorder or drop of network packets inside a transmit buffer that is associated with a **network interface controller** (NIC). This task is performed by the **network scheduler**, which for this purpose uses various algorithms described below.

#### Random early detection [edit]

One solution is to use **random early detection** (RED) on the network equipment's **port queue buffer**.<sup>[8]</sup><sup>[9]</sup> On **network equipment** ports with more than one queue buffer, **weighted random early detection** (WRED) could be used if available.

RED indirectly signals to sender and receiver by deleting some packets, e.g. when the average queue buffer lengths are more than e.g. 50% (lower threshold) filled and deletes **linearly more** or (better according to paper) **cubical more** packets,<sup>[10]</sup> up to e.g. 100% (higher threshold). The average queue buffer lengths are computed over 1 second at a time.

#### Robust random early detection (RRED) [edit]

The **robust random early detection** (RRED) algorithm was proposed to improve the TCP throughput against denial-of-service (DoS) attacks, particularly low-rate denial-of-service (LDoS) attacks. Experiments have confirmed<sup>[11]</sup> that the existing RED-like algorithms are notably vulnerable under Low-rate Denial-of-Service (LDoS) attacks due to the oscillating TCP queue size caused by the attacks.<sup>[12]</sup> RRED algorithm can significantly improve the performance of TCP under Low-rate Denial-of-Service attacks.<sup>[12]</sup>

#### Flowbased-RED/WRED [edit]

Some network equipment are equipped with ports that can follow and measure each flow (**flowbased-RED/WRED**) and are hereby able to signal to a too big bandwidth flow according to some QoS policy. A policy could divide the bandwidth among all flows by some criteria.

#### Explicit Congestion Notification (ECN) [edit]

Another approach is to use **IP Explicit Congestion Notification** (ECN).<sup>[13]</sup> ECN is only used when the two hosts signal that they want to use it. With this method, a protocol bit is used to signal explicit congestion. This is better than the indirect packet delete congestion notification performed by the RED/WRED algorithms, but it requires explicit support by both hosts to be effective.<sup>[14]</sup> Some outdated or buggy **network equipment** drops packets with the ECN bit set, rather than ignoring the bit.<sup>[*original research*?]</sup> **Sally Floyd**, one of the authors of ECN has published detailed information on the status of ECN, including the version required for **Cisco IOS**.<sup>[9]</sup>

When a router receives a packet marked as ECN capable and anticipates (using RED) congestion, it sets the ECN flag notifying the sender of congestion. The sender should respond by decreasing its transmission bandwidth, e.g., by decreasing the TCP window size (sending rate) or by other means.

#### Cisco AQM: Dynamic buffer limiting (DBL) [edit]

**Cisco Systems** has taken a step further in the Catalyst 4000 series with engine IV and V. Engine IV and V has the capability to classify all flows as *aggressive* (bad) or *adaptive* (good). It ensures that no flows fill the port queues for a long time. **DBL** can utilize **IP ECN** instead of packet-delete-signalling.<sup>[15]</sup><sup>[16]</sup>

#### TCP window shaping [edit]

*See also: [TCP window scale option](#)*

Congestion avoidance can also efficiently be achieved by reducing the amount of traffic flowing into a network. When an application requests a large file, graphic or web page, it usually advertises a "window" of between 32K and 64K. This results in the server sending a full window of data (assuming the file is larger than the window). When there are many applications simultaneously requesting downloads, this data creates a congestion point at an upstream provider by flooding the queue much faster than it can be emptied. By using a device to reduce the window advertisement, the remote servers will send less data, thus reducing the congestion and allowing traffic to flow more freely. This technique can reduce congestion in a network by a factor of 40.<sup>[*citation needed*]</sup>

#### Backward ECN (BECN) [edit]

*Backward ECN* (BECN) is another proposed network congestion mechanism. It uses **ICMP source quench** messages as an already existing IP signalling mechanism to implement a basic ECN

mechanism for IP networks, keeping the congestion notifications at the IP level and requiring no negotiation between network endpoints. Effective congestion notifications can be propagated to transport layer protocols, such as TCP and UDP, for the appropriate adjustments in their operations.<sup>[17]</sup>

**Side effects of congestive collapse avoidance** [\[edit\]](#)

**Radio links** [\[edit\]](#)

The protocols that avoid congestive collapse are often based on the idea that data loss on the Internet is caused by congestion. This is true in nearly all cases; errors during transmission are rare on today's fiber-based Internet. However, this causes **WiFi**, **3G** or other networks with a radio layer to have poor throughput in some cases since wireless networks are susceptible to data loss due to interference. The TCP connections running over a radio based **physical layer** see the data loss and tend to believe that congestion is occurring when it isn't and erroneously reduce the data rate sent.

**Short-lived connections** [\[edit\]](#)

The slow-start protocol performs badly for short-lived connections. Older **web browsers** would create many consecutive short-lived connections to the web server, and would open and close the connection for each file requested. This kept most connections in the slow start mode, which resulted in poor response time.


To avoid this problem, modern browsers either open multiple connections simultaneously or **reuse one connection** for all files requested from a particular web server. However, the initial performance can be poor, and many connections never get out of the slow-start regime, significantly increasing latency.

**See also** [\[edit\]](#)

- [Bandwidth management](#)
- [Bufferbloat](#)
- [Cascading failure](#)
- [Choke exchange](#)
- [Erlang unit](#)
- [Max-min fairness](#)
- [Sorcerer's Apprentice Syndrome](#)
- [TCP congestion avoidance algorithm](#)
- [Teletraffic engineering](#)
- [Thrashing](#)

**References** [\[edit\]](#)

- ↑ (Al-Bahadili, 2012, p. 282) Al-Bahadili, H. (2012). *Simulation in computer network design and modeling: Use and analysis* [↗](#). Hershey, PA: IGI Global.
- ↑ TCP Tunnels: Avoiding Congestion Collapse (2000) [↗](#)
- ↑ Van Jacobson, Michael J. Karels. *Congestion Avoidance and Control* [↗](#) (1988). *Proceedings of the Sigcomm '88 Symposium*, vol.18(4): pp.314–329. Stanford, CA. August, 1988. This paper originated many of the congestion avoidance algorithms used in TCP/IP.
- ↑ [RFC 2001](#) [↗](#) - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms
- ↑ [RFC 2581](#) [↗](#) - TCP Congestion Control
- ↑ [RFC 3390](#) [↗](#) - TCP Increasing TCP's Initial Window
- ↑ TCP Congestion Avoidance Explained via a Sequence Diagram [↗](#)
- ↑ <sup>a</sup> <sup>b</sup> Sally Floyd: RED (Random Early Detection) Queue Management [↗](#)
- ↑ Sally Floyd, Van Jacobson. *Random Early Detection Gateways for Congestion Avoidance* [↗](#) (1993). *IEEE/ACM Transactions on Networking*, vol.1(4): pp.397–413. Invented Random Early Detection (RED) gateways.
- ↑ An Analytical RED Function Design Guaranteeing Stable System Behavior [↗](#) Quote: "...The advantage of this function lies not only in avoiding heavy oscillations but also in avoiding link under-utilization at low loads. The applicability of the derived function is independent of the load range, no parameters are to be adjusted. Compared to the original linear drop function applicability is extended by far...Our example with realistic system parameters gives an approximation function of the cubic of the queue size..."
- ↑ (PDF) <http://sites.google.com/site/cwzhangres/home/files/RREDRobustREDAlgorithmtoCounterLow-rateDenial-of-ServiceAttacks.pdf?attredirects=0> [↗](#). **Missing or empty |title= (help)**
- ↑ <sup>a</sup> <sup>b</sup> Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen, RRED: Robust RED Algorithm to Counter Low-rate Denial-of-Service Attacks [↗](#), IEEE Communications Letters, vol. 14, pp. 489-491, 2010. *Ref* [↗](#)
- ↑ [RFC 3168](#) [↗](#) - The Addition of Explicit Congestion Notification (ECN) to IP
- ↑ Comparative study of RED, ECN and TCP Rate Control (1999) [↗](#)
- ↑ Active Queue Management [↗](#)
- ↑ Enabling Dynamic Buffer Limiting [↗](#)
- ↑ A proposal for Backward ECN for the Internet Protocol [↗](#)

- "Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice" by John Evans, Clarence Filsfils (Morgan Kaufmann, 2007, [ISBN 0-12-370549-5](#))
- [RFC 2914](#) - Congestion Control Principles, Sally Floyd, September, 2000
- [RFC 896](#) - "Congestion Control in IP/TCP", John Nagle, 6 January 1984
- Introduction to *Congestion Avoidance and Control* , Van Jacobson and Michael J. Karels, November, 1988



---

## Books [\[edit\]](#)

- "Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice" by John Evans, Clarence Filsfils (Morgan Kaufmann, 2007, [ISBN 0-12-370549-5](#))

---

## External links [\[edit\]](#)

- Nagle, J. [RFC 896](#) - *Congestion control in IP/TCP internetworks* (1984)
- Floyd, S. [RFC 2914](#) - *Congestion control principles* (2000)
- Floyd, S. and K. Fall, *Promoting the Use of End-to-End Congestion Control in the Internet* [\[IEEE/ACM Transactions on Networking, August 1999\]](#)
- Sally Floyd, *On the Evolution of End-to-end Congestion Control in the Internet: An Idiosyncratic View*  (IMA Workshop on Scaling Phenomena in Communication Networks, October 1999) (*pdf format*)
- Linktionary term: Queuing [\[edit\]](#)
- Pierre-Francois Quet, Sriram Chellappan, Arjan Durrresi, Mukundan Sridharan, Hitay Ozbay, Raj Jain, " Guidelines for optimizing Multi-Level ECN, using fluid flow based TCP model" [\[edit\]](#)
- Sally Floyd, Ratul Mahajan, David Wetherall: RED-PD: RED with Preferential Dropping [\[edit\]](#)
- A Generic Simple RED Simulator for educational purposes by Mehmet Suzen [\[edit\]](#)
- Approaches to Congestion Control in Packet Networks 
- Papers in Congestion Control [\[edit\]](#)
- Random Early Detection Homepage [\[edit\]](#)
- Explicit Congestion Notification Homepage [\[edit\]](#)
- TFRC Homepage [\[edit\]](#)
- AIMD-FC Homepage [\[edit\]](#)
- TCP congestion control simulation: Fast recovery [\[edit\]](#)

- [TCP congestion control simulation: fast recovery](#)
- [Recent Publications in low-rate denial-of-service \(DoS\) attacks](#)

Categories: [Network performance](#) | [Teletraffic](#) | [Transport layer protocols](#) | [Technological problems](#) | [Packets \(information technology\)](#)

This page was last modified on 3 September 2015, at 13:26.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) | [About Wikipedia](#) | [Disclaimers](#) | [Contact Wikipedia](#) | [Developers](#) | [Mobile view](#)

