ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό
Πανεπιστήμιο Αθηνών

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ &
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Advanced Network Architectures (M132)

Assignment 1

Data flow and congestion control principles.

Name:	**Evangelos Siatiras**
e-mail:	**EN2190001@di.uoa.gr**

Purpose of flow control in data networks

1.1 Purpose of Flow Control

As in details described in [1], n most networks, there are circumstances in which the externally offered load is larger than can be handled even with optimal routing. Then, if no measures are taken to restrict the entrance of traffic into the network, queue sizes at bottleneck links will grow and packet delays will increase, possibly violating maximum delay specifications. Furthermore, as queue sizes grow indefinitely, the buffer space at some nodes may be exhausted. When this happens, some of the packets arriving at these nodes will have to be discarded and later retransmitted, thereby wasting communication resources. It is thus necessary at times to prevent some of the offered traffic from entering the network to avoid this type of congestion. This is one of the main functions of flow control. Flow control is also sometimes necessary between two users for speed matching, that is, for ensuring that a fast transmitter does not overwhelm a slow receiver with more packets than the latter can handle.

1.2 Means of Flow Control

Generally, a need for flow control arises whenever there is a constraint on the communication rate between two points due to limited capacity of the communication lines or the processing hardware. Thus, a flow control scheme may be required between two users at the transport layer, between a user and an entry point of the subnet (network layer), between two nodes of the subnet (network layer), or between two gateways of an interconnected network (internet layer). There are many approaches to flow control, including the following:

- **Call blocking.** Here a session is simply blocked from entering the network (its access request is denied). Is Happening to a session only after a failed negotiation of some "service contract," for example, an agreement on some service parameters for the session's input traffic (maximum rate, minimum rate, maximum burst size, priority level, etc.)
- **Packet discarding.** When a node with no available buffer space receives a packet, it has no alternative but to discard the packet.
- **Packet blocking.** When a packet is discarded at some node, the network resources that were used to get the packet to that node are wasted. It is thus preferable to restrict a session's packets from entering the network if after entering they are to be discarded.
- **Packet scheduling.** In addition to discarding packets, a sub-network node can exercise flow control by selectively expediting or delaying the transmission of the packets of various sessions

1.3 Main Objectives of Flow Control

We look now at the main principles that guide the design of flow control algorithms. Our focus is on two objectives:

- **Limiting delay and buffer overflow.** For important classes of sessions, such as voice and video, packets that are excessively delayed are useless and may be given high priority.
- **Fairness.** When offered traffic must be cut back, it is important to do so fairly as complicated, as it is by the presence of different session priorities and service requirements.

Network performance under congestion

As mentioned above one of the problems must be solved with flow control is related with the whole network and is so called “Prevent the network Congestion”. The network starts get congested, a lot of delays emerging with result of having data loss or packet drop. Eventually these data has to be retransmitted. Once, the data have been resending the problem is increased. This is explained in this graph. On small traffic load, there is a small number of packet per second that are arriving in a node or in a receiver (end-to-end systems) in a certain rate which can been handled perfectly until reaching the first knee. This increases linear, whatever comes into the system can perfectly been handled and sent out. This is the best-ideal system until that knee. However, by limiting the traffic flow is not that efficient as we can achieve a higher throughput than the limit. At the knee we start reaching a situation where we have enough traffic but we can still handle it. Meaning that probably we can go a bit higher in the graph (after the knee).Summing up, in the ideal case (with infinite buffers, without protocols and retransmissions) we definitely can reach the max throughput and we cannot do nothing better that this upper limit. So in the ideal case the whole system is managed to reach the max throughput, so the data will stay in the system for ever due to infinite buffers, so they will ever reach their destination. So this system is ideal only for the node or the overall system but not for the services depending on the data. Referring to the congestive collapse, we start continuously pushing data in the system and as the traffic load increases without controlling the flow (without limiting the amount of data continuously pushed). Such that way the system starts losing data with result of start having a lot of delays in the data delivery. So it starts having retransmissions. Since we have retransmissions, we cannot count them to actual throughput. The whole system is starting to collapse as is filled out with retransmissions and the throughput goes to zero. So overall we managed to have a very high traffic load but the actual throughput to zero. The purpose of flow control is to keep the throughput close to max. The flow control curve tends to extend a little beyond the knee and stops at specific point always lower to the ideal case (infinite buffers) with the goal to be as close as possible to ideal.

Flow control approaches to deal with congestion in data networks

Flow control is a multi-layered approach as it is applied to different layers, different technologies. Each layer can handle flow control independently from the other layers. Respective to the type of technology is performed to the related layer. A littler layer type of technology, layer 2 approach is so called "ARQ-Automatic Repeat Request". A higher layer approach, at layer 4 is referred as "Flow Control in TCP". Layer 4 is end-to-end, so it does not get involved with the flow control and traffic regulation inside the links but it focusing to their outputs. The goal is to have the max possible utilization can be achieved.

In general, we can divide congestion control [2] mechanisms into two broad categories: **open-loop congestion control** (prevention) and **closed-loop congestion control** (removal).

- Open Loop Congestion Control: In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination.
    - **Retransmission Policy** Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.
    - **Window Policy** The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-N window for congestion control. In the Go-Back-N window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.
    - **Acknowledgment Policy :** The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.
    - **Discarding Policy :** A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.
    - **Admission Policy :** An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual- circuit connection if there is congestion in the network or if there is a possibility of future congestion.
- **Closed-Loop Congestion Control** Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols.

- **Back-pressure:** The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source.
- **Choke Packet** A choke packet is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke packet methods. In back-pressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly.
- **Implicit Signalling** In implicit signalling, there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the network from other symptoms.
- **Explicit Signalling** The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signalling method, however, is different from the choke packet method. In the choke packet method, a separate packet is used for this purpose; in the explicit signalling method, the signal is included in the packets that carry data. Explicit signalling can occur in either the forward or the backward direction**.**
    - **Backward Signalling** A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.
    - **Forward Signalling** A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

Flow control and congestion control in the Internet protocol

The Transmission Control Protocol (TCP) [3] is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP).

4.1 Purpose of TCP Flow Control

The hosts on each side of a TCP connection each set aside a receive buffer for the connection. When the TCP connection receives bytes that are correct and in sequence, it places the data in the receive buffer. The associated application process will read data from this buffer, but not necessarily at the instant the data arrives. Indeed, the receiving application may be busy with some other task and may not even attempt to read the data until long after it has arrived. If the application is relatively slow at reading the data, the sender can very easily overflow the connection's receive buffer by sending too much data too quickly. TCP thus provides a flow control service to its applications by eliminating the possibility of the sender overflowing the receiver's buffer. Flow control is thus a speed matching service – matching the rate at which the sender is sending to the rate at which the receiving application is reading.

4.2 Flow Control Mechanism

TCP provides flow control by having the sender maintain a variable called the receive window. Informally, the receive window is used to give the sender an idea about how much free buffer space is available at the receiver. In a full-duplex connection, the sender at each side of the connection maintains a distinct receive window. The receive window is dynamic, i.e., it changes throughout a connection's lifetime. Let's investigate the receive window in the context of a file transfer. Suppose that host A is sending a large file to host B over a TCP connection. Host B allocates a receive buffer to this connection; denote its size by RcvBuffer. From time to time, the application process in host B reads from the buffer. For the flow control mechanism implementation we define the following variables:
LastByteRead is the number of the last byte in the data stream read from the buffer by the application process in B and LastByteRcvd is the number of the last byte in the data stream that has arrived from the network and has been placed in the receive buffer at B.

Because TCP is not permitted to overflow the allocated buffer, we must have:
LastByteRcvd − LastByteRead <= RcvBuffer
The receive window, denoted RcvWindow, is set to the amount of spare room in the buffer:
RcvWindow = RcvBuffer − [LastByteRcvd − LastByteRead]
Because the spare room changes with time, RcvWindow is dynamic.The connection use the variable RcvWindow to provide the flow control service. Host B informs host A of how much spare room it has in the connection buffer by placing its current value of RcvWindow in the window field of every segment it sends to A. Initially host B sets RcvWindow = RcvBuffer. Note that to pull this off, host B must keep track of several connection-specific variables. Host A in turn keeps track of two variables, LastByteSent and LastByteAcked, which have obvious meanings. Note that the difference between these two variables, LastByteSent − LastByteAcked, is the amount of unacknowledged data that A has sent into the connection. By keeping the amount of unacknowledged data less than the value of RcvWindow, host A is assured that it is not overflowing the receive buffer at host B. Thus host A makes sure throughout the connection's life that :LastByteSent − LastByteAcked ≤ RcvWindow
There is one minor technical problem with this scheme. To see this, suppose host B's receive buffer becomes full so that RcvWindow = 0. After advertising RcvWindow = 0 to host A, also suppose that B has nothing to send to A. As the application process at B empties the buffer, TCP does not send new segments with new RcvWindows to host A – TCP will only send a segment to host A if it has data to send or if it has an acknowledgment to send. Therefore, host A is never informed that some space has opened up in host B's receive buffer: host A is blocked and can transmit no more data! To solve this problem, the TCP specification requires host A to continue to send segments with one data byte when B's receive window is zero. These segments will be acknowledged by the receiver. Eventually the buffer will begin to empty, and the acknowledgements will contain non-zero RcvWindow.

Internet protocol performance under flow/congestion control

5.1 Performance of Internet protocol when flow/congestion control is implemented

TCP congestion control algorithms will be more conservative than might be predicted by the model, leading to measured throughput around a third less than estimated [4]. Loss is determinant because loss events regularly cap cwnd, the sender will never send more than W bytes at every RTT, effectively limiting its maximum throughput. Also, for a small loss rates dramatically affect TCP throughput, even at low values. The Mathis model is particularly appropriate to estimate the TCP throughput in network paths with regular packet loss, which happens in a steady state when there’s a constant rate of inelastic cross traffic in the path, as might be seen in heavily loaded data center environments. If no loss event is observed, cwnd steadily increases until a corrupted packet is detected, or the receiver advertises an insufficient receive buffer.

5.2 TCP UDP Comparison

UDP in contrast with TCP does not provide flow control [5]. For a typical UDP implementation, UDP will append the segments (more precisely, the data in the segments) in a finite-size queue that "precedes" the corresponding socket (i.e., the door to the process). The process reads one entire segment at a time from the queue. If the process does not read the segments fast enough from the queue, the queue will overflow, and segments will get lost.
TCP congestion control [6] regulates an application's transmission rate via the congestion window mechanism. Many multimedia applications do not run over TCP for this very reason – they do not want their transmission rate throttled, even if the network is very congested. Many Internet telephone and Internet video conferencing applications typically run over UDP. These applications prefer to pump their audio and video into the network at a constant rate and occasionally lose packets, rather than reduce their rates to "fair" levels at times of congestion and not lose any packets.

5.2 UDP throughput in the presence of TCP

Taking into consideration the cumulative effect of the datagrams loss, the throughput decreases. According to the paper in [7] If sending rate of UDP is reduced, it hardly matters for UDP throughput, as TCP follows congestion algorithm which will grab any bandwidth available till congestions occurs. TCP has greedy congestion control mechanism. There is variation of UDP throughput for different TCP fractions. Maximum UDP is throughput is when there is no TCP.

References

[1] S. Thombre, “Modelling of UDP throughput,” 02 2017. [Online]. Available: https://www.researchgate.net/publication/315765956_Modelling_of_UDP_throughput.

[2] K. W. R. a. J. F. Kurose, “Connection-Oriented Transport: TCP,” 2000. [Online]. Available: https://www.net.t-labs.tu-berlin.de/teaching/computer_networking/03.05.htm.

[3] K. W. R. a. J. F. Kurose, “TCP Congestion Control,” 2000. [Online]. Available: https://www.net.t-labs.tu-berlin.de/teaching/computer_networking/03.07.htm.

[4] J. Antunes, “Validating A Very Simple Model for TCP Throughput,” 24 06 2013. [Online]. Available: https://blog.thousandeyes.com/a-very-simple-model-for-tcp-throughput/.

[5] “Congestion Control,” [Online]. Available: http://www.idc-online.com/technical_references/pdfs/data_communications/Congestion_Control.pdf.

[6] D. Bertsekas and R. Gallager, “Data Networks,” [Online]. Available: https://web.mit.edu/dimitrib/www/Flow_Control_Data_Nets.pdf.