M.I.T. Laboratory for Computer Science

Request for Comments No. 249
May 3, 1983

A Survey on Congestion Control in Computer Networks

by Lixia Zhang

1. Introduction

In any transmission network there exists a potential danger of congestion if a proper traffic control mechanism has not been installed. The same is true for packet-switched computer communication networks, and this has drawn significant attention from network builders and researchers. Over the years, however, despite much effort that has been put on the problem, it still remains as a further research area.

What is the exact problem to be solved? And why is it so hard? How much have we done in approaching the right solution, or how much have we learned from attacking the problem? What seems to be a right direction to try our further effort? This paper attempts to give some thought in answering these questions.

Section 2 gives a clear definition of congestion control, along with a more detailed description of the problem, which, hopefully, answers the first and second questions. Section 3 summarizes previous works on congestion control for answering the third question. Finally, Section 4 tries to answer the last one.

2. Description of the Problem

2.1. Definitions of congestion control and flow control

There are many different definitions of congestion control and flow control that often cause confusion. Some people use the two words interchangeably, others apply them to different meanings. For example, Kleinrock defined flow control as follows: "the problem of flow control is to regulate the rate at which data crossed the boundary of the communication subnetwork (both into and out of the network)." [4], while Tanenbaum said that "Flow control is an end-to-end phenomenon, whereas congestion control deals with problems occurring at intermediate IMPs as well." [11] Therefore it is necessary to first state our understanding and definitions precisely.

We denote the word ***flow control*** to mean the data transmission rate control at each end-to-end connection level, because packet streams flow along such logical connections, as water flowing through a pipe. Flow control concerns the destination host which controls the source sending rate to protect itself from overflow. It is not concerned with problems of resource availability that may arise within intermediate subsystems interposed between the source and destination.

In a network, resources are shared by all the hosts attached to it. Transmission rate restriction imposed by the network, due to the finite amount of available resources, is called ***congestion control.*** It is the network that controls hosts' sending rates to avoid jamming the network.

2.2. Separating flow control from congestion control

From above definitions we see that both flow and congestion controls may result in restricting the transmission rates of source hosts. So it might be desirable and feasible to consider joining congestion control and flow control mechanisms together in order to simplify implementation, as sometimes when designing a family of protocols for a homogeneous network. We should keep in mind, however, that congestion control and flow control are two different things for different purposes. Both have the same effect on source host transmission rates, that is, on the performance seen by network users, but based upon different requirements from the network and the destination host, respectively. An immediate conclusion that can be drawn is: it may be possible to consider the two jointly, but one cannot be substituted by the other.

If considering flow control and congestion control separately, flow control depends only on the

available resources at the destination host. Because of this one-to-one nature, and also because people have much experienced with resource management in a single system, the problem is comparatively easier and indeed has been solved in practice, whereas congestion control is not so fortunate. This paper will focus attention on the congestion control issue alone and ignore flow control. That is not to say, however, that there is no relation between the two at all. (See a companion paper "Congestion Control at Internet Gateway" for some investigation on this. The first conclusion coming out is that flow control would like to have information from congestion control, but no need for the other way around.)

2.3. Where the problem came from

Congestion control has been well recognized as a resource sharing problem [4, 8]. What are the resources being shared? Compared to the old circuit switched communication networks, such as telephone networks, which do not contain either data processors or storage, the fundamental differences of a packet switched network are the *asynchronously time-sharing of communication bandwidths* as well as the installation of data processors at each switching node, hence bringing increased functionality into the network, such as routing, error checking, failure detection, etc. Storage was added at the same time to buffer the temporarily waiting packets due to the contention for use of shared resources. Processing power and communication bandwidths are surely valuable resources being shared. Buffer pool, as a necessary tool having been brought into computer networks, was probably not one of the primary resources in the goal of sharing communication utilities - it is waiting queue space, just as space is the first need for building a parking lot, but cannot help with jamming on the highway at all.

In short, we want the resources of a computer network to be asynchronously shared by all hosts in the net, i.e. to meet user requests with unpredictable request time, transmission rate, and transfer duration from all over the network. Any physical resource is finite and need be managed for sharing. That is our definition of congestion control.

2.4. Why the problem is hard

Consider two different kinds of networks: local area networks, which usually are fully connected through single transmission media (e.g. a bus or a ring), and store-and-forward networks (e.g. a long haul network or LANs connected through gateways). The dynamic sharing problem exists in both

cases. Because of having cheap and abundant communication bandwidths, neglectable delay, and more importantly, the single central resource, local area networks have solved the problem fairly well by setting down well formed sharing rules, such as TOKEN scheme in ringnet and CSMA-CD in Ethernet. A store-and-forward network, on the other hand, is a set of distributed resources. As pointed out by Kleinrock [4], the problem in the store-and-forward network is much harder due to the following facts:

1. Resources are geographically distributed and cannot be easily managed on dynamic demands without reducing the sharing. For example, one data transfer may require resources available at several switch nodes.

2. Because of this geographically distributed environment, there exists unknown communication delay between users(hosts) and resources, which causes synchronization and consistency problem.

3. Control information consumes valuable network resources.

4. Reliability problem in this geographically distributed environment.

In other words, the geographical distribution of resources is the main cause of the hardness.

3. Previous Works on Congestion Control

Summarizing the congestion control mechanisms that have been implemented in networks (or have been proposed to be implemented, excluding mathematical solutions.)[1] they may be divided into three basic categories:

1. Buffer management schemes.

2. Virtual circuit based control.

3. Source choking mechanism.

These practical schemes rely upon the local observation of some critical resource usage that is presumed to be a sensitive indicator of the network load. All known studies but one picked buffer space as the only critical resource for the measuring. CIGALE, a French experimental network, investigated a mechanism based on bandwidth usage (see below), and no one seems to have considered processor as a potential bottleneck yet. We describe each kind briefly in the following.

---

[1] I also left the well-known Isarithmic scheme [1], the first devised congestion control scheme, which has never been implemented.

3.1. Buffer management used in congestion control

In current store-and-forward networks, available buffer pool at each switch node usually is small. Various ways have been investigated on how to allocate this very limited space to different packet classes [5, 9]. The classification may correspond to inbound/outbound lines or to packet "age", i.e. the number of hops the packets have traveled in the network. When the buffer space used reaches allowed limit, newly arrived packets are discarded. For example, both Arpanet and DECnet set limitations on the minimum number of buffers that must be allocated to every inbound/outbound lines as well as on the maximum length of a single outbound queue [2]. DECnet also uses a favoring transit-over-input algorithm[2]. Buffer management methods are widely used in both datagram and virtual circuit networks.

This kind of schemes are a very passive mechanism for congestion control, if they can be called a control mechanism at all. Notice that all buffer schemes can only apply to the networks with the hop-by-hop acknowledgment/retransmission mechanism. Under the assumption that every sending node will keep a copy of outstanding packet until the acknowledgment comes back from the receiving node, no matter what kind of buffer limitations being used, the "control" actually relies on the fact that congestion occurring at any node in the route will eventually push upstream back to the source host to stop or slow it down. Although the favoring transit-over-input algorithm may partly shut down incoming traffic, which is questionable in terms of fairness between "junior" and "senior" packets, the source host that caused congestion will not be informed until the congestion wave creeps back, by which time the whole path is buffer-full. If the routing procedure has tried to spread out the congestion, considerable part of the network might be filling up, resulting in long delay with possible impact of degrading traffic in other directions. If a network protocol has no acknowledgment/retransmission procedure and simply drops packets when buffers are full, there will be no way to stop the source host that causes congestion. Another criticism to this technique is the waste of communication bandwidths due to internodal retransmission - each node along the congested route keeps retransmitting again and again until congestion disappears, or finally believes the successive neighbor "died" and gives up.

Worse yet, notice that when the interface between the host and the network is locked up due to

---

[2]When a switching node gets congested, it rejects new packet transmission from immediate host(s) but tries to forward transit packets in order to save the network resources already spent on the transit packets.

congestion, it is the lowest protocol layer that runs into trouble. In most designs, there is no protocol specification or implementation dealing with this kind of situation, i.e. how to pass the congestion information bottom-up to let higher layers properly handle it. The higher layers probably just sit and wait until timeout, without knowing what exactly happened. Even assuming the lower layer could pass some information, what can it say to the higher layer? It knows nothing about the network situation either but the shutdown of the interface.

Nevertheless, presently buffer-based congestion control is the most popular and available scheme for datagram networks (but see source choke below). The main reason for this is that a datagram network has no traffic information at all, except seeing each individual packets. Buffer schemes, as well as the source choke mechanism that will be described later, work on a packet-by-packet basis. Because of the difficulties mentioned in section 2.4, so far there is no close coordination between a datagram network and the hosts attached to it. Hosts look the network as an abstract transmission tool that they can use at will. Although people have learned the lessons that "one must view packet communications as a system rather than a trivial leased line substitute" [4], so far the change of the view has not been seen in protocol design or implementation as far as the interface between user requests and the network is considered. Once any problem occurs in the network, either congestion or failure, the only detection tool available to the host is a timer. But the timeout can only tell that *something* goes wrong, not *what* it is. The host may or may not do the right thing in handling the timeout, e.g. it may try retransmission in the case that the timeout was caused by network congestion. If the timer is set long, more outgoing traffic might have been passed from applications to the transmission protocol before the timeout, all buffered in the system. On the other hand, networks, especially datagram networks, do not really have *control* over data traffic, if we understand control as "to exercise authority or dominating influence over, to direct, to regulate"(Heritage dictionary), rather than simply dropping packets or only trying to recover from disasters. A successful congestion control mechanism should match incoming traffic to the handling capacity of the network and make congestion very unlikely to occur.

3.2. Virtual circuit based control

For virtual circuit networks, two ways are usually considered available for congestion control. One is resource reservation: storing the information of connection states as well as required bandwidth or buffer space of each virtual connection at every intermediate node. The

establishment of a virtual connection implies dedication of resources of one form or another along the network path. Some networks have an upper limit on the number of VCs which can be opened simultaneously, such as SNA of IBM systems [3]. Generally every VC network has some sort of network load measurement. New connection requests will be rejected when the network becomes saturated, in contrast with the datagram network which does not conservatively refuse new traffic entering the network (because it cannot tell) but will discard packets midway wherever it gets congested.

Few VC network implementations handle bandwidth reservations directly (Tymnet does [10]), more often they convert the requests to buffer reservation. In either case, the amount of resources reserved is usually an upper limit, because the accurate requirements are difficult or impossible to know. The packet-switch node therefore usually oversells itself to achieve better throughput. But when some VCs are really reaching the peak of traffic fluctuation, the network performance degrades considerably and the service to users may become intolerable or even disconnected.

It is well known that burstiness is an essential nature of computer network traffic. A virtual connection may remain for a period of time, e.g. a remote login connection can stay for hours, whereas the real data traffic sent by the connection, as the sum of bunch of bursts separated by relatively long quiet periods, is possibly at second-level only. Hence we see that the virtual-connection based resource reservation is not a right solution to congestion problem. This kind of resource reservations resembles in some way the old circuit switched network in which network resources are dedicated to each connection for an entire session. It is likely to result in either wasting resources if doing conservative reservation or hardly achieving satisfiable performance if being too greedy.

The second way is to convert network congestion control problem to each virtual connection flow control. When congestion occurs, a VC network may show its advantage of using the control on each connection and slowing down packet flows, but this does not come for free. Simply using acknowledgment delay to control traffic is a tricky game. It increases the uncertainty between the two ends (because everyone uses timer to check whether the partner is still alive). It also implies that retransmission timers must be set long enough in order not to inject more duplicates into the network when the packets were received correctly but the acknowledgments are delayed. If the control is done by using hop-by-hop dynamic window on each connection, it is a considerable task to keep track with, due to the large number of virtual connections.

After all, we should not ignore the basic arguments about the relative advantages of datagram and VC networks. Being a packet-switched network, the only mandatory function is packet transmission, of course, with all necessary supporting functions such as routing, congestion control, fragmentation, or even accounting for commercial networks. All other functions, e.g. sequencing, error recovery, etc., are application dependent and should be left open. It cannot be known in advance whether a particular application will require, or will not be willing to pay the cost of, those functions. Network users may also desire to have choices between tradeoffs of delay, throughput, cost, reliability, and so on. In this aspect, a datagram network offers a flexible service basis upon which higher layers can build other functions as needed. A VC network, on the other hand, is more expensive and restrictive,less robust (cautious users always build end-to-end check and recovery themselves, duplicating the effort supposed to be done by the network), and does not meet the requirements of all applications. Nevertheless, it does have more control over every virtual connection, if this is considered as an advantage in some cases.

3.3. Source choking scheme

Only two network protocols implemented direct source choking mechanism for congestion control. One is IP [7] (see [12] for analysis). The other is French network CIGALE, of which a distinct characteristic is measuring the network load by the average utilization percentage of transmission lines instead of buffer space [6], as all the other networks do. Every node periodically tests the status of all its outbound lines[3] and exchanges this information in routing update messages with neighbors. When the percentage is over a given threshold, e.g. 70%, it is considered to be an indication of possible congestion, and the line is put in warning state. A choke packet will be sent back to the source host for every packet going through that line to request the reduction of traffic submitted to that direction. If the line gets more heavily loaded, e.g. over 80%, all packets heading to that line will be dropped at the network entry nodes. This is somehow similar to IP source quench algorithm, but in addition, it also attempted to recognize the true critical resources in the network, in its case are bandwidths, and to measure their status directly. However, it missed the case that CPU becomes a bottleneck, while buffer utilization measure, though indirectly, can find the situation in both CPU and bandwidth bottleneck cases.

---

[3]The lines speeds are 48kbps, 19.2kbps, and 4.8kbps. This measuring has more overhead than measuring buffer usage.

It is interesting to compare this direct source choke scheme with the other two congestion control mechanisms. Although IP source quench and CIGALE's algorithm are different in many details, they both approached an essential point in congestion control. That is, the network should notify and slow down the source hosts that caused congestion as soon as possible, in an attempt to keep excess traffic *out of* the network, whereas the previous two controls, either buffer-based or VC-based, basically let excess traffic fill the path and then rely on their eventually controlling by hop-by-hop back pressure.

The remaining problems in source choke scheme are

~ Both IP and CIGALE are datagram network protocols and lack data traffic information, thus causing the following problems

* They have to work on a packet-by-packet basis. So a stream of choke messages are likely to be sent to the source hosts when congestion occurs. This is a big waste of communication bandwidths, as well as being the waste of processing power of switching nodes and annoying to the hosts.

* The choke message is no more than a simple congestion notice. It does not really tell the host what to do. The host has to try out itself: first slow down some, wait for a while, then may speed up again, so on and so forth.

~ They ignored (or missed?) the effect of unknown, varying communication delay between the network and hosts. This delay plus the fact that hosts have to blindly adjust their transfer rates (to meet the restriction as well as not to waste bandwidths if the congestion goes away soon) can easily cause traffic oscillation.

4. Seeking New Direction for Congestion Control

4.1. Critical resources in the network

What are the bottlenecks limiting the transmission rate and causing congestion? As argued earlier, processing power and communication bandwidths are basic sharing resources in the network. It is easy to see that these two are the potential causes of congestion at switch nodes. Whenever the CPU speed cannot catch up with the input rate to process all incoming packets promptly, or any of the output lines cannot drain out all packets routed to it at the required rate, packets will accumulate at the node, queuing up at either input lines or output lines. If the burst is short and there is enough space to hold it, the node can drain out all the packets soon. Otherwise congestion occurs.

From the last section, however, we see that the attention is almost exclusively focused on buffer space problem in solving congestion control, away from what we think here.

4.2. Why people paid so much attention to buffering

At the time the packet-switched network was first built up, the memory of minicomputers, which were installed at each switch node, was expensive and the size was pretty small. This lack of memory caused a number of problems. First, when congestion occurs, either caused by CPU bottleneck or line bandwidths, the by-product is buffer overflow. Second, the small buffer pools at switching nodes are really easy to fill out by any nontrivial size message transfer or a few simultaneous users, even the traffic just fluctuates a little with no real danger of congestion. Third, the small size of available buffer space at the switch node probably limited people's imagination of doing any meaningful traffic control on a dynamic basis. Buffer-full causes the dropping of new coming packets, which in turn results in retransmission, wasting communication bandwidth that is considered to be the most expensive resource in the network. The idea of retransmission probably was first thought of for error/failure recovery, but it is now largely used for congestion recovery, since people have not succeeded in preventing congestion from happening. If someone likes to investigate how much percentage of retransmission is caused by congestion in a particular network, it would be an interesting result to know.

4.3. The effect of buffering in congestion control

Buffer space plays a dominant role in many proposed congestion control algorithms. It was assumed that if buffer space was reserved at each intermediate node, congestion would be avoided. If we put enough buffer space at each node, will the congestion problem go away? A switch node must have buffer space available to accept an incoming packet, but what we really want is to forward the packet towards its destination as soon as possible. How long does the packet have to wait until being forwarded? Buffer reservation mechanisms usually cannot answer this question. In fact, buffer management, no matter how good it is, can only avoid *deadlock*, but not *congestion*. A limited buffer pool at the switching node may cause considerable dropping of packets and degrade the effective network throughput. A huge buffer space, on the other hand, can cause manifold delay, making higher layers harder to implement, or even being intolerable to the application.

We can look at the problem in another way by using a limit analysis method. We see that either

infinitely fast CPU or infinitely fast output lines will make the corresponding potential bottleneck disappear (actually need not be infinitely fast, it is a function of the sum of input line rates), while infinite buffer space cannot solve the problem if any of the other two performs at a lower rate than required. In contrast, if both of them are fast enough, there will be no need for buffer space.

The above argument is not to say, however, that we do not need buffer space in congestion control. The randomness and burstiness of data traffic are well-known although we do not have a suitable statistical model for it. Since we cannot predict the rapidly changing load of the network and we must also take into consideration of the inherent communication delays between the network and hosts, enough buffer space is necessary to hold sudden packet bursts before the control information gets back to source hosts. We also need buffer space to smooth down the transmission between lines with different speeds - we cannot request the fast line to transmit at the exact speed of slow line but that the two average rates must match.

4.4. What the key is to the solution of congestion control

Now the rapidly developing VLSI technology has brought us the luxury of considerably larger size of memory with low cost, we need no longer worry about the severe space restriction at the switching node if it is built of a modern microprocessor. But rather, we should now pay our attention to how to use this large buffer space in solving congestion. If we still use it in the conventional way, i.e. no restriction on the traffic until the pool is used up, serious delay will occur because it is now possible for an waiting queue to grow considerably long.

In looking for new direction to solve congestion problem, we have learned from past the following things

- ~ Excess traffic must be kept out of the network as far as possible. Since requiring reservation in advance is not acceptable, congestion detection may use dynamic information abstracted from current traffic, and keeping excess traffic outside may need to be on a relative long-term basis.

- ~ To achieve this, congestion control should handle hosts sending rates directly and quickly, as source choking algorithm approached, instead of playing buffer games in the network.

- ~ In order to do any meaningful control, the network must have information about dynamic data traffic. Otherwise its control attempt is hard to accomplish, as we saw

above. Current datagram networks have no knowledge of traffic, while VC networks have static (compared to data traffic) connection information. Both are not proper bases for congestion control.

~ Processing power and/or communication bandwidths are crucial resources and decide the performance of the network. Attention should be focused on how to control traffic based on their limits, avoiding waste, such as unnecessary retransmissions.

In a companion paper, "Congestion Control at Internet Gateway", above thought is used to design a new congestion control algorithm in a specified environment, internet with IP implementation.

References

[1]

D. Davies.
The Control of Congestion in Packet Switched Computer Networks.
IEEE Transactions on Communications, June, 1972.

[2]

Transport functional specification.
1980.
AA-j059A-TK

[3]

J. Gray and T. McNeill.
SNA Multiple-system Networking.
IBM System Journal 18(2), 1979.

[4]

L. Kleinrock.
Principles and Lessons in Packet Communications.
IEEE Proceedings, November, 1978.
This paper is a good summarization of difficulties encounted in the network control - resource sharing in a distributed environment.

[5]

L. Kleinrock.
Flow Control: A Comparative Survey.
IEEE Transactions on Communications, April, 1980.
The paper summarizes the current state of the art, but a missing aspect is that there is no analysis on what effects those control mechanisms have on the performance seen by or offered to the network users - it looked the problem purely from the network point of view.

[6]
J. Majithia, M. Irland, J. Grange, N. Cohen, C. O'Donnell.
Experiments in Congestion Control Techniques.
Proceedings of the International Symposium on Flow Control in Computer Networks, 1979.
No measurement on total control overhead.

[7]
J. Postel.
Internet Control Message Protocol.
Technical Report RFC 792, DARPA, September, 1981.

[8]
L. Pouzin.
Flow Control in Data Networks - Methods and Tools.
?, 1976.

[9]
L. Pouzin.
Methods, Tools, and Observations on Flow Control in Packet-Switched Data Networks.
IEEE Trans. on Communications, April, 1981.
A large portion of this paper resembles the previous one, giving the impression that the field had not been advanced much during the five year period.

[10]
J. Rinde & A. Caisse.
Passive Control Techniques for Distributed Networks.
Proceedings of the International Symposium on Flow Control in Computer Networks, 1979.

[11]
Andrew S.Tannenbaum.
Computer Networks.
Prentice Hall, 1981.

[12]
Lixia Zhang.
Congestion Control at Internet Gateway.
Technical Report RFC 250, MIT-LCS-CSR, May, 1983.