

Autonomous Congestion Control in Delay-Tolerant Networks

Scott Burleigh, Esther Jennings
{Scott.Burleigh, Esther.Jennings}@jpl.nasa.gov

ABSTRACT

Congestion control is an important feature that directly affects network performance. Network congestion may cause loss of data or long delays. Although this problem has been studied extensively in the Internet, the solutions for Internet congestion control do not apply readily to challenged network environments such as Delay Tolerant Networks (DTN) where end-to-end connectivity may not exist continuously and latency can be high. In DTN, end-to-end rate control is not feasible. This calls for congestion control mechanisms where the decisions can be made autonomously with local information only. We use an economic pricing model and propose a rule-based congestion control mechanism where each router can autonomously decide on whether to accept a bundle (data) based on local information such as available storage and the value and risk of accepting the bundle (derived from historical statistics). Preliminary experimental results show that this congestion control mechanism can protect routers from resource depletion without loss of data.

1. INTRODUCTION

In recent years, there has been a strong interest in the architecture and protocol design for challenged network environments where continuous end-to-end connectivity may not exist and/or round-trip latency can be high. Some examples of such Delay Tolerant Networks (DTN) are the interplanetary network, mobile tactical military networks, and rescue/response networks. In graph theoretic terms, DTN is modeled as a directed multi-graph where the link capacities are time varying [1].

The success of the Internet depends on many factors. An important one is the end-to-end congestion control mechanism provided by TCP that prevents congestion collapse. The main idea behind TCP's end-to-end congestion control is for each source to be sensitive to resource depletion in the network, to determine the supportable rate of data injection into the network. An end-to-end TCP ACK is used to notify a source of data arrival so that the source can pace its data injection rate accordingly (sources are *self-clocking*).

However, this technique cannot be applied to DTN directly because we cannot assume continuous end-to-end paths in the network. Due to potentially intermittent connectivity, we need to design a congestion control mechanism that does not depend on end-to-end dialogue. Intuitively, the congestion control decisions should be made autonomously at each router using local information only.

Therefore, we propose to use an economic model and a rule-based congestion control mechanism that relies only on local state information. The main challenge is to autonomously make local (non-cooperative) decisions to achieve network performance that maximizes the successfully delivered information value.

2. BACKGROUND

The performance of a network degrades when congestion occurs. Congestion is caused by heavy traffic load; it results in data loss due to buffer exhaustion at routers or long delays in data transmission due to retransmission in response to data loss.

Numerous congestion control mechanisms have been developed for the Internet as well as for wireless networks. The optimization criteria for these mechanisms may be high throughput and low latency, or high information (in bits) per unit energy. In [2], Keshav advanced a theoretical basis for congestion control mechanism design. He suggested that the key issue was efficient resource allocation and proposed that an economic model can be used to derive the congestion control mechanism based on resource availability. Similar pricing methods are used by Heikkinen [3] and Siris, Briscoe and Songhurst [4].

We present an alternative economic model and devise simple rules for each router to make local decisions autonomously (in non-cooperative fashion) to avoid network congestion in the DTN. The general network performance objectives we are concerned with are high throughput (information value) and low delay.

The definitions of flow control and congestion control can be found in widely used networking textbooks [5, 6]. *Flow control* is needed to ensure the destination can handle all of the incoming data from the source; this mechanism controls the rate of traffic injection by applications into a network. *Congestion control* is needed when buffers in routers are oversubscribed; this mechanism minimizes data loss within a network due to buffer space limitations. Congestion control indirectly induces flow control, but flow control may also be induced on a pairwise basis in a network that is not congested.

A. Flow Control and Congestion Control in the Internet

In the Internet, continuous end-to-end connectivity is expected; this enables instantaneous rate matching where transmission rate is always expected to be equal to the arrival rate. Any violation of this expectation (e.g. excessive transmission rate at a source causing an excessive arrival rate which results in rapid depletion of buffer space) will trigger immediate corrective action (flow control). Since the signal propagation delays are low in the Internet, any required corrective action can immediately be performed at the source of the traffic.

Flow control in the Internet is handled end-to-end. TCP at the destination detects growth in its buffer space occupancy signifying that the destination application is receiving (from TCP, via a socket) at a slower rate than the data arrival rate at TCP. TCP at the destination responds by reducing the acknowledgement (ACK) rate, which will be detected by TCP at the source. Then TCP at the source will reduce its transmission rate. This imposes *flow control at the source application* (the source socket) that indirectly relieves the excessive growth rate in buffer space occupancy at the destination.

Congestion control is handled from router-to-source. Heavy traffic load causes instantaneous growth in buffer space occupancy at a router – that is, the router is transmitting at a slower rate than the data arrival rate. Two solutions are available:

- *Explicit*: the router sends an ICMP source quench packet to the source, resulting in reduced TCP transmission rate. As above, the reduced TCP transmission rate imposes flow control at the source application.
- *Implicit*: the router discards datagrams, causing absence of TCP ACK, which causes TCP at the source to detect reduced acknowledgement rate, again causing TCP at the source to reduce its transmission rate and imposing (once again) flow control at the source application.

B Flow Control and Congestion Control in DTN

In DTN, continuous end-to-end connectivity cannot be expected: any single point-to-point link may be disabled at any moment. Instantaneous rate matching is not expected and instantaneous growth in buffer space occupancy is not anomalous. Only sustained net growth in buffer space occupancy is anomalous, and such anomalies are not trivially detected binary phenomena: how long is “sustained”?

Unlike TCP, DTN protocols include no end-to-end acknowledgement mechanism that can be the vehicle to perform the corrective actions at the traffic source immediately. The acknowledgement available is a hop-by-hop custody transfer acknowledgement (CT-ACK, either custody acceptance or custody refusal). Even this cannot solve the congestion problem because signal propagation delays may be high, and the urgent corrective action cannot wait for CT-ACK; it must be performed *locally* and *autonomously*:

- *Flow control can only be local*: sustained net growth in buffer space occupancy at the source, for whatever reason, must result in the imposition of *flow control on the source application*.
- *Congestion control can only be local*: sustained net growth in buffer space occupancy at the router, however caused, cannot be solved by *explicit congestion control* because a source quench message might be irrelevant by the time it arrives at the source host. The only plausible option seems to be *implicit congestion control*: a router discards a bundle due to resource depletion, causing absence of custody acceptance (and, eventually, arrival of a custody refusal CT-ACK), which causes sustained net growth in buffer space occupancy at the current custodian (the upstream router), which in turn causes that bundle agent to discard bundles due to resource depletion, etc.; ultimately the source bundle agent detects sustained net growth in its own buffer space occupancy, resulting in imposition of flow control at the source application.

3. METHOD

In short: congestion control in bundling as in the Internet is accomplished by inducing flow control at the applications that are the source of the excess traffic and, since that flow control is driven by sustained net growth in buffer space occupancy, a natural way to implement DTN congestion control is to propagate buffer utilization stress back through the network to the source bundle agents. We accomplish this by declining to take custody of bundles, forcing the sending Bundle agent to retain the bundles and thereby increasing that agent's local demand for buffer space, forcing it in turn to refuse custody of bundles, and so on.

Given this strategy, the remaining question is how to determine when to decline to take custody of a bundle in order to conserve local buffer space prudently. This is where we apply a *financial* model of buffer space management. The basic notion is that (a) unoccupied buffer space is regarded as analogous to money and (b) routing network traffic is regarded as analogous to the daily financial activities of an investment banker.

A. Financial Model

A router has limited buffer space, analogous to the fixed amount of capital a banker has to work with.

Notionally, we imagine that the application that owns the sender and receiver of a bundle will pay a *conveyance fee* to get the bundle delivered. The *fee* will be a function of the bundle size and the quality of service requested; the *fee* is not a function of the number of forwarding hops. The *banker* (router) will receive a *commission* for completing one hop of each bundle's end-to-end route, and the commission will be a percentage of the bundle's total conveyance fee.

Accepting custody for an inbound bundle for forwarding equates to a banker *buying* a non-interest-bearing debenture: the bundle is acquired at the cost of certain amount of free buffer space.

The bundle's TTL corresponds to the due date on the debenture: the banker knows that, in the worst case, he will recover his investment capital (buffer space) when the TTL expires. However, the banker's rate of compensation – *commissions* – is a function of his rate of conveyance activity or *churn* (traffic, throughput). That is, the router's incentive is to accept the largest possible volume of traffic (number of bytes of bundle payload, weighted by QoS) per unit of time. As the router's resources are limited, in order to convey a large volume of traffic the router must move the traffic (find "buyers" for it) as rapidly as possible so as to free up his investment capital for new purchases. That is, the constraint on the activity of the router is in essence "cash flow", which limits his ability to accept ("purchase") new bundles.

There is a negative incentive to accept custody of bundles with large TTLs, i.e., there is a higher risk that this bundle may tie up the buffer resource for a long time because one cannot predict how fast this bundle will sell (be forwarded). The risk is the chance of experiencing the worst-case scenario, in which the router never manages to forward the bundle – the banker never finds a buyer for the debenture – and is forced to hold it to maturity. This could crowd out the router's ability to accept other bundles that are potentially more marketable and therefore reduce his compensation.

B. Rule-Based Congestion Control Mechanism

In Bundling, we implement negative *purchasing decisions* by refusing to accept bundles for forwarding. Refusing a bundle that is flagged for custodial forwarding constitutes an explicit custody refusal, which results in transmission of a *Custody refused* message.

As a bundle *banker*, one freely uses one's capital to *purchase* bundles until the remaining capital balance is reduced to a level that makes one uneasy, and at that point one starts to economize: one spends less readily on high-risk bundles. When one's available capital increases (when there is income or, in Bundling terms, when one's outbound bundles begin to be accepted by downstream forwarding nodes so that one's buffer space can be reclaimed), one's uneasiness begins to diminish and one tends to take a little more risk in buying, knowing that one making purchases for which one is more confident there is a market.

Let N = the number of bytes occupied by a given bundle.

Let Q = the current number of bytes (aggregating all bundles occupying buffer storage) that are queued up for transmission at the router.

Let K = the router's maximum capacity of all bundle buffers, in bytes.

Rule 1: if $(Q + N) > K$, refuse the bundle.

Otherwise, consider the residual TTL and buffer occupancy net growth as follows to decide whether to reject the bundle.

The *residual TTL* (RTTL) of a bundle, R , is the remaining number of seconds before the bundle expires, i.e., before the bundle may be discarded from buffers even if custody is accepted.

Net growth in buffer space occupancy over a given interval is the sum of the sizes of all bundles accepted (inserted into buffer space) over that interval minus the sum of the sizes of all bundles successfully forwarded (removed from buffer space) over that interval.

Assume that records are kept regarding net growth in buffer space occupancy such that M_T , the mean net growth per second over the most recent T seconds, can be computed for any value of T .

Let G_R be the projected net growth in buffer occupancy over interval R ; we compute it as

$$G_R = R * M_R.$$

The worst-case total number of bytes D of bundle payload that might have to be refused due to acceptance of custody of a given bundle is computed as

$$D = (G_R + Q + N) - K.$$

That is, accepting custody of a bundle has a cost (a worst-case opportunity cost) only if $D > 0$, i.e., if available buffer space might completely fill before the bundle is either forwarded or discarded.

Rule 2: if $D \leq 0$, then you must accept the bundle (the projected usage of buffer space is less than the total buffer capacity).

If $D > 0$, then we must compute the bundle's acceptance risk in order to decide whether to accept this bundle and potentially have to refuse custody of some future bundle(s) or,

alternatively, refuse this bundle so that we can accept future bundles of potentially higher value.

Let F_q be the conveyance fee per byte that is charged upon delivery of a bundle whose QoS is q ; then the total fee P that will be charged upon delivery of a bundle of length N and QoS q is given by

$$P = (N * F_q).$$

The *value* of accepting a given bundle is some function of (i.e., is the commission on) the fee that will be charged upon delivery of the bundle, P . That is, the router (banker) has positive incentive to accept custody of bundles, with a bias toward bundles of high QoS.

Assume that records are kept regarding the aggregate value of bundles accepted in the past such that V_T , the mean value accepted per second over the most recent T seconds, can be computed for any value of T .

The *risk* of accepting a given bundle is the worst-case number of byte-seconds of buffer space that may be consumed by this bundle. It is simply the product of the bundle's size, in bytes, and its residual TTL.

Assume that records are kept regarding the aggregate risk of bundles accepted in the past such that R_T , the mean risk accepted per second over the most recent T seconds, can be computed for any value of T .

The *risk rate* of a given bundle is its risk divided by its value. The mean risk rate over interval T is the mean risk R_T for that interval divided by the mean value V_T for the same interval.

Rule 3: if the risk rate of a given bundle exceeds the mean risk rate over the bundle's residual TTL, then the bundle is of above-average risk; refuse it. Otherwise, accept the bundle.

C. Discussion

The congestion control algorithm is always applied, whether the inbound bundle is flagged for custody transfer or not. If the bundle is non-custodial and the congestion decision is to refuse it, then the bundle is simply lost; this is exactly analogous to UDP datagram loss in a congested IP-based network.

When the decision is to accept the bundle and the bundle is custodial, a custody acceptance message is sent back to the current custodian (nominally, the proximate sender). Reception of this message causes the bundle to be removed from the custodian's buffers, reducing buffer space occupancy and net buffer space occupancy growth, and thus relieving congestion at the custodian. This is somewhat analogous to the function of the ACK in TCP.

When the decision is to *refuse* the bundle and the bundle is custodial, a custody refusal message (an explicit NAK) is sent back to the current custodian. Reception of this message triggers retransmission of the bundle (potentially on a different route and/or hopefully at a time when the proximate recipient is less congested and better able to receive the bundle). At the same time, the absence of a custodial acceptance message – whether the custodial refusal message arrives or not – triggers congestion control:

congestion pressure remains unrelieved because the custodial bundle remains in the custodian's buffers, causing buffer space occupancy and net buffer space occupancy growth to increase over time.

Note that this is unlike TCP/IP, which normally relies on timer expiration (an implicit NAK) to trigger both segment retransmission (potentially on a different route and/or hopefully at a time when the proximate recipient is less congested and better able to receive the bundle) and also congestion control (exponential backoff). Timer-based custodial retransmission is also possible in Bundling but is problematic: it is still not clear how to compute a reasonable custodial timeout interval in a network characterized by highly variable round-trip times.

Routers, motivated to maximize throughput with bias toward high-QoS bundles, refuse bundles as they deem necessary – i.e., in response to congestion. Their storage resources are protected from over-subscription.

When custodial bundles are refused by a router, current custodians upstream of the router that are themselves routers experience increased buffer occupancy – they become congested themselves, triggering an increased likelihood of bundle refusal. Custody refusal therefore effects congestion control by propagating congestion back along the path to the original sender.

When the original sender's bundle agent experiences congestion – i.e., its buffer space begins to be oversubscribed – rate control reduces the rate at which new bundles are issued, and the congestion eventually abates.

Note that a bundle with higher QoS not only has greater value than one with a lower QoS, it also will be forwarded sooner than one with low QoS and is therefore more likely to get forwarded before a given time. This means that its end-to-end delivery latency is likely to be lower. While this is beneficial in itself, it also means that the sender can declare a marginally lower TTL for the bundle upon originating it, without diminishing likelihood of delivery. Since a lower TTL always means a lower residual TTL at every point in the end-to-end path, higher QoS thus tends to further reduce the likelihood of refusal and therefore increase the likelihood of delivery.

The combination of reduced delivery latency and enhanced likelihood of delivery justifies “charging” more for bundle transmission as a function of QoS.

Note also that as a bundle progresses along its end-to-end route, its residual TTL decreases. This too diminishes its acceptance cost (by reducing G_R) and its risk, reducing the likelihood that a router will refuse to accept it. This aligns precisely with our desire to avoid wasting the amount of transmission energy invested to date in a bundle by letting it be discarded late in its path when it is close to being delivered to its destination.

4. EXPERIMENTAL RESULT

In a controlled laboratory environment, we have tested the congestion control mechanism in a simple scenario. The experiment setup consists of three computers, each being a bundle agent. We will refer to them as agents A, B, and C. The convergence layer is

TCP. The topology of the network is a line connecting A to B, and B to C; there is no direct connection from A to C (Figure 1). Bundle agent A has a *driver* application that injects bundles into the network as rapidly as possible in a continuous tight loop destined to a *sink* application at bundle agent C. Flow control at the source is the only mechanism that prevents immediate buffer collapse at A. Each bundle has a size of 60,000 bytes; it is sent with *custody transfer* requested, with a TTL of 1 hour. Since there is no direct connection between A and C, all the bundles are routed through custodial router B.

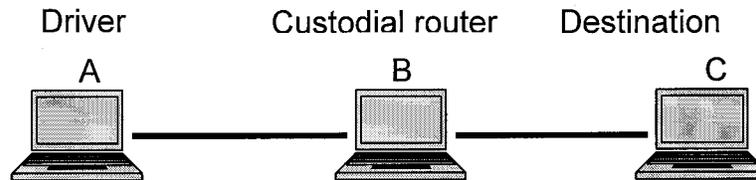


Figure 1. Experiment setup with three bundle agents.

To test our congestion control mechanism, we artificially impose congestion by slowing down bundle agent C, adding forcing a delay of 0, 1, 2, or 4 microseconds per byte before accepting each bundle. The resulting bundle injection rate at A is then measured. For this initial experiment, we did not incorporate disruption of the links.

When running the experiment without activating the congestion control mechanism, we observed congestion collapse at router B, as expected. Running the same experiment with activated congestion control mechanism, we observed no congestion collapse at router B and the measured throughput was 32 Mb/sec.

5. CONCLUSIONS AND FUTURE WORK

In this preliminary study, we proposed a simple autonomous local congestion control mechanism for DTN. In our experiments, the congestion control mechanism is functioning as expected, effectively minimizing data loss within the network due to finite buffer space. The merit of this congestion control mechanism is that each router only needs local information to make custodial decisions autonomously; it does not add any communication overhead to gather network information for decision-making, and it is not subject to failure due to loss of connectivity or large variations in signal propagation latency.

For future work, we intend to run extensive experiments to further validate the congestion control mechanism and evaluate its performance varying different network parameters. As a first study, our economic model is simple; investigation of more sophisticated modeling or algorithms may lead to improved network performance. Another possible trade-study is whether additional information will enable routers to make better decisions to achieve improved network performance, and what would be the overhead cost for getting these information.

6. REFERENCES

- [1] S. Jain, K. Fall and R. Patra, "Routing in a Delay Tolerant Network", in Proceedings of SIGCOMM 2004, Portland, Oregon, Aug. 30 – Sep. 3, 2004.
- [2] S. Keshav, "A Mechanism for Congestion Control in Computer Networks", unpublished preprint, 1989,
<http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/89/mechanism.pdf>.
- [3] T. M. Heikkinen, "On Congestion Pricing in a Wireless Network", *Wireless Networks* 8, 347–354, 2002.
- [4] V. A. Siris, B. Briscoe, and D. Songhurst, "Economic Models for Resource Control in Wireless Networks", in Proceedings of IEEE PIMRC 2002, Lisbon, Portugal, Sep. 15 – 18, 2002.
- [5] D. Bertsekas and R. Gallager, "Data Networks", Prentice Hall, 1987.
- [6] L. L. Peterson and B. S. Davie, "Computer Networks, A Systems Approach", Morgan Kaufmann Publishers, 2nd Edition, 2000.