3rd Canadian Conference on Broadband Research
(CCBR'99), Nov. 1999

Traffic Management for Aggregate IP Streams

Alan Chapman

Nortel Networks
P.O.Box 3511, Station C, Ottawa, Ontario K1Y 4H7
achapman@nortelnetworks.com

H.T.Kung

Harvard University
Division of Engineering and Applied Sciences
33 Oxford Street, Cambridge, Mass. 02138
kung@harvard.edu

ABSTRACT

The IP networking industry is defining ways to differentiate service levels and provide contracted quality of service. Much of the work, such as multi-protocol label switching, is moving toward traffic engineering models for aggregate streams that closely parallel those of ATM networks. A general architecture, similar to that used for ATM connections, is suggested for the traffic management of IP aggregates.

ATM networks use resource management cells for control. These cells are injected, monitored and removed by the network. They do not intrude on the user traffic and do not have to be aware of the user data protocols. Such a control overlay, which is absent from TCP/IP, is well-suited for a future IP network.

The control overlay has particular value for the management of controlled network bandwidth sharing between aggregate IP streams. Issues with TCP, the primary traffic sharing mechanism in today's network, are discussed. An approach that uses TCP's congestion control to manage sharing between aggregated streams of traffic is described. Enhancements are suggested for improving scalability and response time for the TCP control.

Congestion detection in IP network routers has historically been done by monitoring buffer-fill which has some inherent latency. For a flow-controlled aggregate stream it is possible to detect congestion by monitoring link occupancy and thus minimise the latency.

INTRODUCTION

IP networks are evolving and there is a growing need for some demonstrable or contractual qualities of service (QoS). To a large extent the requirements are similar to those of ATM (Asynchronous Transfer Mode) networks and involve control over attributes such as loss, delay, throughput and availability. However, the requirements will most likely be applied to traffic aggregates rather than individual flows.

Why aggregates?

An aggregate traffic stream is a collection of IP flows that are grouped together for common treatment between two points in a network. We call these aggregates **IP trunks**. Packets from all flows in a trunk travel the same path and are subject to the same traffic management policies. The concept of IP trunks brings the Layer-2 values of ATM into the IP network. A well-known example is the Multi-Protocol Label Switching (MPLS) [1] activity in the Internet Engineering Task Force (IETF), although an IP transport, that uses IP addresses as labels, may be appropriate for virtual private networks [2].

By encapsulating many IP flows with a common label, MPLS treats an aggregate traffic stream rather than individual flows. These aggregates can follow determined paths and be given a consistent QoS treatment. A labelled path can exist between any two points in the network and multiple paths can themselves be aggregated within another label. This hierarchical aggregation simplifies the management of network resource and facilitates engineering of QoS.

Performance commitments from a carrier to its customers are likely to be at the aggregate level rather than for individual flows. For applications such as virtual private networks the performance of the aggregate in terms of throughput, latency, loss rates, security and availability over the complete trunk path are more relevant than knowledge about individual flows or network links.

Why congestion control for aggregates?

The Transmission Control Protocol (TCP) is the most widely used method to achieve elastic sharing between end-to-end IP flows. At present the core network basically relies on end-system TCP to provide congestion control and sharing but this will not be acceptable in the future for several reasons:

To avoid time-out, each TCP connection requires that some number of packets be stored in the network and most of that storage occurs in the router buffer at a point of congestion. Without sufficient storage, time-outs will give end users poor performance and prevent any predictable sharing. Unfortunately, providing sufficient storage for a very large number of connections will cause too much latency. As a corollary, if latency is to be contained then the number of connections must be severely limited. Suppose that it is desired to bound the delay in a core network to 10ms. Then, for the minimum possible window of a single one kilobyte packet, the connection cannot run slower than 1Mb/s without going into time-out. This implies that to provide fair sharing of a 1Gb/s link, the number of competing TCPs should be no more than 1,000.

Most traffic today is carried by very short TCP flows. The fast retransmission feature [10] of TCP requires a window size of several packets in order to work and short flows often never achieve a large enough window size to be able to recover from a packet loss without entering time-out. Fine-grain bandwidth sharing, where sharing is achieved over time intervals under 1 or 2 seconds, is important for interactive applications, but is not possible unless connections avoid time-out.

TCP is a dominant protocol today but may not be so in the future, especially as the amount of traffic using User Datagram Protocol (UDP) is growing rapidly. It is dangerous, therefore, to rely only on host TCP to achieve good sharing in the future network.

End system TCP implementations are not totally standard and, in particular, can be modified to the benefit of the end system but at a cost to the network.

Finally, end-to-end TCP will strive for fair sharing between individual flows and this will not meet the requirements in a network managed as aggregate streams. For example, for those Internet Service Providers which provide services to institutions, the management of minimum throughput and the control of elastic bandwidth sharing will usually be more relevant at the aggregate level than at the level of individual flows.

Congestion control at the aggregate level also reduces the number of flows visible to the core network. This protects user flows from the effects of highly populated IP network [8].

It is beneficial to the network operator to aggregate many user flows into a new, preferably lossless, flow-controlled stream between chosen edge-points in the network. This pushes the loss and latency problems out to the edge of the network where it is possible to be more selective in allocating that loss and latency to individual IP flows. Further, a network-based method for providing controlled, elastic bandwidth-sharing between these aggregates will be essential to ensure full use of the network without detracting from promised performance.

MANAGEMENT OF IP TRUNKS

Historically, the IP network management has considered end-to-end flows and router-to-router links. Given that IP trunks will be such an important part of the future network architecture, there is need for a well designed and consistent approach to their traffic management. There must be ways to provide monitoring and management of these trunks in order to check the integrity of the path as well as testing how well it is meeting the QoS requirements.

In a multi-protocol network, the management architecture should be independent of the end users protocols. In ATM networks the use of resource management cells has been well developed. These cells are injected into the traffic stream in a non-intrusive way in order to test various attributes of the connection.

A similar approach of injecting management packets is suitable for the management of trunks in an IP network and allows the management to be over any defined path in the network rather than over single links or between end systems. The decoupling from user data packets and protocols means that the management packets can be inserted as often as the application needs and can carry information specific to that application between the two ends of the trunk.

For example, management packets can be inserted into the trunk to test for continuity of the path. The far end of the trunk responds to these packets to confirm that the path is working. The absence of response from the other end of the trunk can be used to trigger the diversion of trunk traffic to a back-up route in far less time than it would take by waiting for the routing protocols to propagate the information. In a path equipped for differentiated services [9], the acknowledgements can be protected against loss with packet marking to make failure detection more robust.

Similarly, loss can be sampled by sending numbered packets which are acknowledged by the far end. Latency can be sampled by injecting packets with time-stamps. GPS (global positioning system) deployment can make these latency measurements very accurate.

USING TCP FOR ELASTIC SHARING BETWEEN IP TRUNKS

In addition to monitoring performance, management packets can be used to facilitate controlled elastic sharing. At present there is no mechanism for elastic sharing between aggregates but this is essential for keeping network utilisation high without detracting from trunk performance commitments. It would be desirable to reuse TCP for this purpose because of the large body of knowledge available and also because the buffer management mechanisms [3, 7] in routers exist for, and are being evolved for, TCP.

There are, however, some aspects of TCP that one would want to avoid for trunks:

It is not practical to encapsulate the user data in TCP headers. This adds bandwidth and processing overhead for every packet, and would make it difficult to manage trunks within trunks.

It is not desirable to incur the overheads (storage, re-ordering and latency) of retransmission for user data. Apart from the cost of extra storage at the sender and receiver, this feature would incur latency for all flows in the trunk without any knowledge as to whether the data sources prefer reliability to lower latency. It is better to leave reliability to the individual end-user protocols.

It is not desirable to incur the loss of user data as is usual when TCP probes for available bandwidth. It is better to incur loss closer to the edge of the network where management or protocols can be more selective in allocating that loss to chosen flows.

What is wanted from TCP is:

The end-to-end simplicity. Any complexity of TCP is contained within the sender and receiver and very little is required from the intermediate switches.

The robustness of the self clocking nature of TCP. The number of packets in the network is contained in the sense that new packets cannot be injected until old ones leave.

The tested approach of additive increase and multiplicative decrease in rate. This has been shown to provide predictable bandwidth-sharing for long lived flows.

We define a **TCP trunk** as an IP trunk whose data packets are subject to TCP's congestion control mechanism but without the undesirable features [2, 5]. This is achieved by means of a decoupled management flow [5, 11] in the following manner.

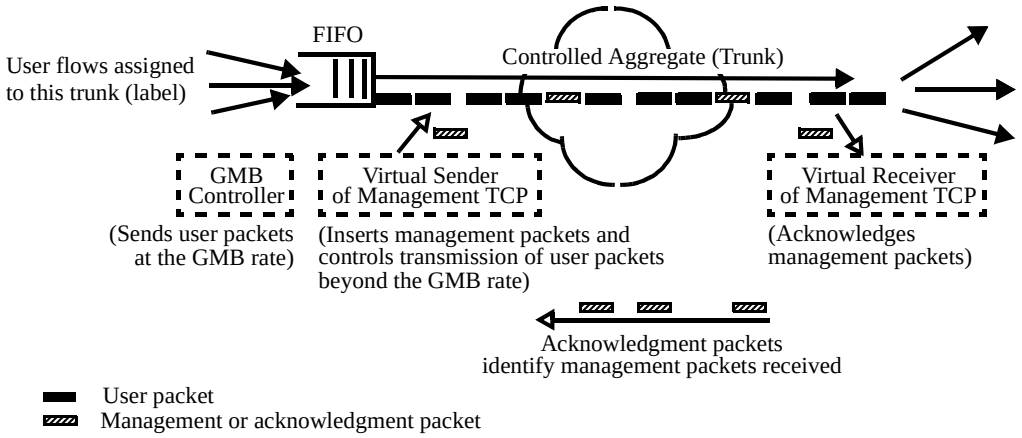


Figure 1 : Use of management TCP and GMB controller in implementing TCP's congestion control and guaranteed minimum bandwidth (GMB) for an aggregated traffic stream

A stream of management or control packets is sent into the trunk by a virtual TCP sender and received and acknowledged by a corresponding virtual TCP receiver at the other end of the trunk (Figure 1). This management TCP connection carries no user data and deals only with a notional stream of bytes. The stream of control packets sample the state of congestion at the routers along the trunk path and, in normal TCP fashion the sender adjusts its sending rate based on the discard or marking of the control packets. However, in addition, the sending rate of the control stream will directly control the sending rate of user data over the trunk.

When the intermediate routers are equipped for differentiated services, loss priority makes it possible to protect the user data and only discard control packets so that a lossless, flow-controlled connection is achieved. See [5] for an analysis of the lossless property, and related experimental results. Moreover, a TCP trunk can traverse an ATM network over a UBR connection and, using packet loss priority, can provide lossless elastic sharing without the need for ATM-based ABR service. (A native TCP trunking service can be also provided directly at the ATM layer; but this topic is beyond the scope of this paper.)

A virtual maximum segment size (VMSS) [5, 11] is defined as the number of user data bytes released for each control packet. This is an average number since the packet boundaries of the user data are maintained. To keep some synchronism between the real data and the sampling of congestion a control packet is only released into the trunk, on average, after one VMSS of user data is sent. For efficiency, if there is no user data to be sent then no control packets are sent and, furthermore, in the continued absence of data, the management TCP will steadily reduce its window down to the minimum.

The release of user data under control of the management TCP is separate from and in addition to the release of data resulting from a configured guaranteed minimum bandwidth (GMB) transmission rate.

OPTIMIZATIONS OF MANAGEMENT TCP

The mechanism of the management TCP is network-based and under control of the network operator. It also operates over a fixed path. It is possible, therefore, to consider changes to the mechanism which are beneficial in this environment.

Speeding up response to congestion

Since in a TCP trunk there is little chance of packet order being changed, it is acceptable to assume packet loss the first time that a duplicate ACK is received, rather than wait for three occurrences as is usual in host-based TCP. This modification will make the trunk respond faster to congestion. If congestion notification is implemented in future routers then the packets are marked rather than discarded and the sender will get an even faster indication of network congestion.

Extending TCP operation into the low-speed domain

As discussed earlier, there is a limit to how slowly TCP connections can run and the absolute limit is one packet per round trip time (RTT). Such small windows are undesirable since the rate of increase of transmission in this case is very large. Increasing a one-packet window to two packets will double the rate, and many connections doing this will almost certainly result in some packet loss. Fast retransmission [10], which helps a connection avoid time-out, can only work when the window contains several packets and good sharing requires that the connections do not enter time-out (see [6]). For these reasons it would be desirable that the window is always above some minimum size and that when bandwidth sharing required the connection to run more slowly, the sender employed a secondary mechanism to achieve this.

One such mechanism is based on artificial increases of RTT at the TCP sender. After receiving an acknowledgement, the sender will delay the release of a packet for some period. The total of the real RTT plus the artificially added time is defined as the virtual round trip time (VRTT)

Doubling the VRTT has the same effect on transmission rate as halving the window size. The additive increase in rate is achieved by decreasing the VRTT. This can be made to directly emulate the rate increase that would be experienced by increasing window size. When the VRTT once again equals the real round trip time then TCP returns to the normal mode of operation.

Avoiding large windows

Buffer management schemes such as random early detection (RED [3]) spread packet loss statistically over all the flows passing through the congestion point. Instead of the sudden and immediate loss experienced by drop tail buffers, the loss will stabilize at the value that maintains an acceptable average buffer-fill.

It is known that, under fast retransmit and recovery of TCP, the window size (W) and the loss ratio (L) are linked by the relationship [4]:

$$L = K/W^2$$

where K is some constant. Thus, the average number of round trip times for a connection to experience a lost packet can be calculated as:

$$\text{Average number of RTTs before a loss} = W/K$$

It takes some time to realise ‘averages’ and typically some connections see no loss for a while and others see more than the average. The effect of multiple rate decreases on one connection is much less than that of single decreases on multiple connections which means greater excursions in buffer-fill. The larger window takes much longer (e.g., many RTTs) to recover after a decrease. Overall, large windows give a looser control loop, wider ranges of buffer-fill and therefore wider ranges of latency.

In a TCP trunk, the amount of data sent per control packet (VMSS) can be configured to a relatively large value to avoid large windows. When there is a large value for VMSS, the ability to extend virtual round trip time (VRTT) will still permit the trunk to run slowly when necessary.

Controlled sharing

Current TCP implementations strive for fair sharing but are influenced by the round trip time (RTT) experienced and maximum segment size (MSS) used. Loss in the network, particularly with buffer management schemes such as random early detection (RED), tend to make all connections have the same window size. The rate that a connection achieves is then directly proportional to MSS/RTT.

In a managed network of TCP trunks it is possible to define a benchmark value for MSS/RTT and to make each connection aware of how its own value compares to the benchmark. By suitably adjusting the base values of VMSS and VRTT or by changing its reaction to loss, the trunk can ensure fair sharing or deliberately unfair sharing with other trunks.

LOW-LATENCY THROUGH VIRTUAL BUFFER ACCOUNTING

Today’s IP networks use buffer-fill as the trigger to begin discarding packets in order to slow down TCP sources. Large buffer-fills produce large latency and, in fact, this has been a benefit in helping the source TCP senders to slow down. The round trip time experienced under congestion conditions can be several times that of the idle network. For TCP trunks using the concept of virtual round trip time, the sender has no limit on how slowly it can run and needs no buffering help from the network. It is possible, therefore, to consider having minimal latency at the congestion points. This can be achieved in the following way.

To minimise latency at the congestion point, the process of discarding packets must begin before the outgoing link becomes fully utilised. In this case there is no buffer-fill and discard must be based on some measure of link utilisation. In order to re-use the buffer management techniques such as random early detection (RED) that are being developed, the concept of a virtual buffer is introduced (Figure 2). This virtual buffer has one token added to it for every packet that arrives but

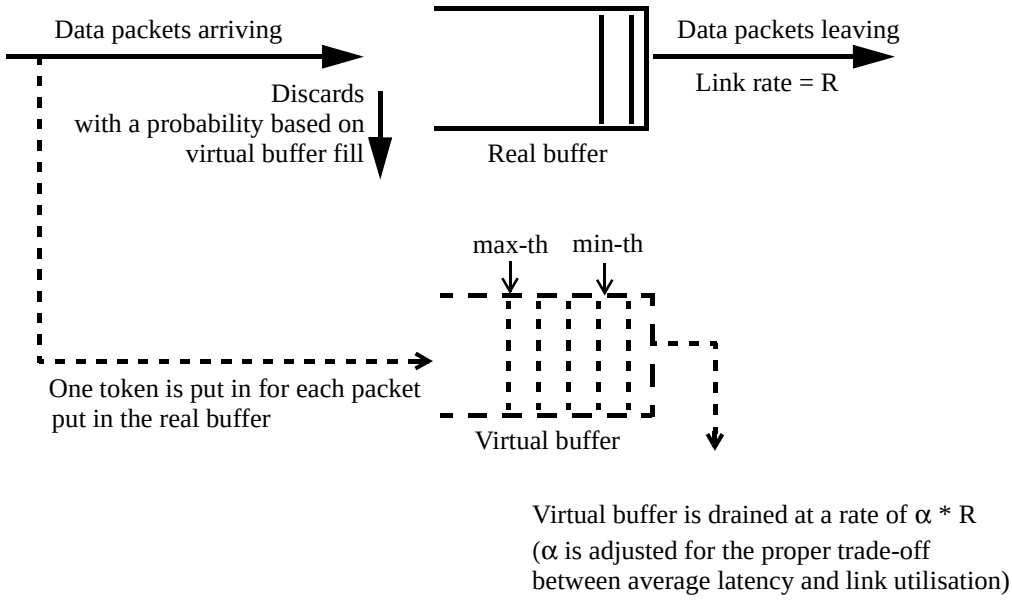


Figure 2. Avoiding high buffer fill, in real buffer, by using a virtual buffer

is drained at a rate less than the real link. The fill of this virtual buffer is used to trigger discard and the drain rate can be adjusted to trade off link utilisation against real buffer-fill.

A network with this feature deployed would allow low-latency trunks, thereby avoiding the need for special queues for real-time traffic and generally giving a crisper performance for interactive applications.

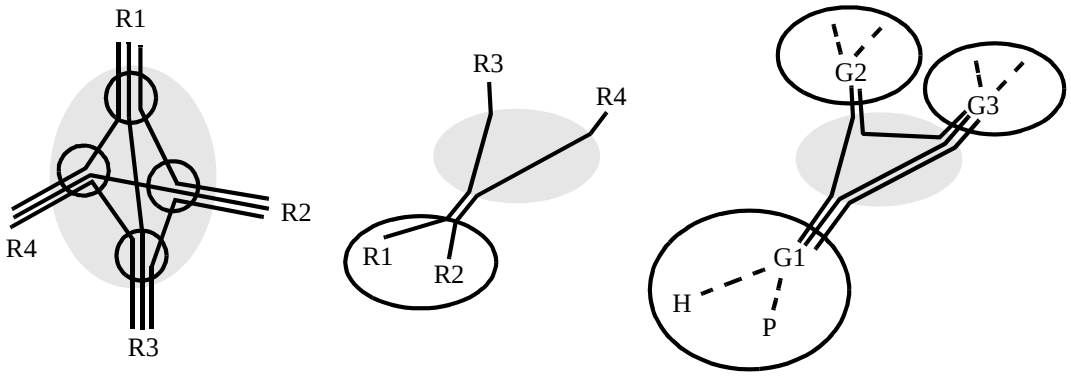
DEPLOYMENT SCENARIOS

This section describes some deployment scenarios of TCP trunking. In these scenarios TCP trunks provide guaranteed and elastic bandwidths in those areas of networks where traffic congestion may occur. Because of its easy deployment, involving only the two end nodes of a trunk to implement the associated management TCP, and its ability to use available bandwidth dynamically, TCP trunking represents an attractive enhancement to other traffic engineering approaches such as those based on ATM, Frame Relay, MPLS and VLANs.

Example scenarios

Three deployment scenarios are illustrated in Figure 3:

- Core Networks: Between each pair of edge routers (R), one or multiple bidirectional pairs of TCP trunks are provided.
- Enterprise Networks: Routers inside an enterprise network are connected via TCP trunks to routers in the other side of the congested network region.
- Virtual Private Networks (VPNs): TCP trunks connect gateways (G), of various sites of a VPN. There are multiple trunks between some sites to differentiate traffic treatments. Behind each gateway, there could be hosts and proxy servers, denoted by H and P, respectively. A proxy server can classify packets based on their IP addresses, protocol types, etc. For each out-going packet, a gateway can select which trunk to use, based on its classification.



(a) Core network (b) Enterprise network (c) VPN

Figure 3. TCP trunking usage examples, for (a) core networks, (b) enterprise, and (c) VPN. Shaded regions denote potential congestion areas in the network. Each solid line denotes a bidirectional pair of TCP trunks. In (a), round circles inside the shaded region denote routers and switches; in (b) and (c) these are omitted for clarity.

Discussions on deployment

Dedicated TCP trunks can be provided for UDP flows so that their total bandwidth usage can be constrained, in spite of the fact that UDP flows do not perform congestion control by themselves. (See [5, 11] for experimental results on the use of TCP trunks to contain UDP flows.) For those campus and backbone networks where use of UDP-based streaming protocols has been increasing rapidly, this could be a major reason for deploying TCP trunks.

Non-trunk traffic can be allowed in a network region where TCP trunks are used. For example, via bandwidth partitioning, TCP trunks may compete only among themselves for a certain portion of network bandwidth, while traffic which is not carried by TCP trunks uses the remaining bandwidth. Of course, there will be no need for bandwidth partitioning if the region has sufficient bandwidth and expects no congestion.

For a network region where bandwidths are abundant, end points of TCP trunks can be anywhere inside the region. That is, they do not have to be on the network edge. For example, Figure 3 (b) assumes that the enterprise network is a high-bandwidth network, and thus, it is appropriate that the trunk end points R1 and R2 are inside the network. Because of trunk end points can be placed anywhere convenient to the user, this flexibility eases the deployment and management of TCP trunks.

An experimental testbed for TCP trunking has been deployed in Taiwan over an 80km ATM link connecting two cities, Taipei and Hsinchu. It is an enterprise model, where the trunk end points are in campuses with gigabit Ethernet links.

Additional testbeds are available at Harvard University for the purpose of validating basic properties of TCP trunking. See [5, 11] for various performance results on TCP trunks.

CONCLUSIONS

It is suggested that a new traffic management architecture is needed for future IP networks, where much of the traffic is managed as aggregates between intermediate points. A general management approach, similar to that used in ATM, has been described.

The particular application to congestion control meets a future need for sharing between aggregates. In addition it protects the user's short flows from time-out and reduces the number of elastic flows seen by the core network and therefore the latency encountered. Having the network congestion control in the network operator's management domain ensures its integrity and permits enhancements and evolution to be independent of the host-based protocol implementations.

It has been shown that the well-tested congestion control mechanisms of TCP can be reused to provide elastic sharing of bandwidth between aggregate streams without the need for changes in the intermediate routers. By varying VMSS and VRTT, TCP trunks can always operate under a moderate sized window to achieve improved robustness and fairness. It has further been shown that, with enhancements to the intermediate routers, the performance can be improved and, in particular, very low-latency trunks can be implemented.

ACKNOWLEDGMENT

This research was partially supported by Nortel, Sprint, Air Force Office of Scientific Research Multidisciplinary University Research Initiative Grant F49620-97-1-0382, and National Science Foundation Grant CDA-94-01024.

REFERENCES

[1] IETF Multiprotocol Working Group, http://www.ietf.org/html.charters/mpls-charter.html

[2] A. Chapman and H. T. Kung, "Enhancing Transport Networks with Internet Protocols," IEEE Communications Magazine, Vol. 36, No.5, May 1998, pp. 100-104

[3] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," Transactions on Networking, 1993

[4] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM'88

[5] H. T. Kung and S. Y. Wang, "TCP Trunking: Design, Implementation and Performance," IEEE ICNP'99

[6] D. Lin, and H. T. Kung, "TCP Fast Recovery Strategies: Analysis and Improvements," IEEE INFOCOM'98, pp. 263-271.

[7] D. Lin and R. Morris, "Dynamics of Random Early Detection," ACM SIGCOMM'97

[8] R. Morris, "TCP Behavior with Many Flows," IEEE ICNP'97

[9] K. Nichols, V. Jacobson and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," IETF draft, 1998

[10] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," RFC 2001, 1997

[11] S. Y. Wang, "Decoupling Control from Data for TCP Congestion Control," Ph.D. Thesis, Harvard University, September 1999