TCP Nice: A Mechanism for Background Transfers

Arun Venkataramani Ravi Kokku Mike Dahlin *

Laboratory of Advanced Systems Research
Department of Computer Sciences
University of Texas at Austin, Austin, TX 78712
{arun, rkoku, dahlin}@cs.utexas.edu

Abstract

Many distributed applications can make use of large *background transfers* — transfers of data that humans are not waiting for — to improve availability, reliability, latency or consistency. However, given the rapid fluctuations of available network bandwidth and changing resource costs due to technology trends, hand tuning the aggressiveness of background transfers risks (1) complicating applications, (2) being too aggressive and interfering with other applications, and (3) being too timid and not gaining the benefits of background transfers. Our goal is for the operating system to manage network resources in order to provide a simple abstraction of near zero-cost background transfers. Our system, TCP Nice, can provably bound the interference inflicted by background flows on foreground flows in a restricted network model. And our microbenchmarks and case study applications suggest that in practice it interferes little with foreground flows, reaps a large fraction of spare network bandwidth, and simplifies application construction and deployment. For example, in our prefetching case study application, aggressive prefetching improves demand performance by a factor of three when Nice manages resources; but the same prefetching hurts demand performance by a factor of six under standard network congestion control.

1 Introduction

Many distributed applications can make use of large *background transfers* — transfers of data that humans are not waiting for — to improve service quality. For example, a broad range of applications and services such as data backup [29], prefetching [50], enterprise data distribution [20], Internet content distribution [2], and peer-to-peer storage [16, 43] can trade increased network bandwidth consumption and possibly disk space for improved service latency [15, 18, 26, 32, 38, 50], improved availability [11, 53], increased scalability [2], stronger consistency [53], or support for mobility [28, 41, 47]. Many of these services have potentially unlimited bandwidth demands where incrementally more bandwidth consumption provides incrementally better service. For example, a web prefetching system can improve its hit rate by fetching objects from a virtually unlimited collection of objects that have non-zero probability of access [8, 10] or by updating cached copies more frequently as data change [13, 50, 48]; Technology trends suggest that "wasting" bandwidth and storage to improve latency and availability will become increasingly attractive in the future: per-byte network transport costs and disk storage costs are low and have been improving at 80-100% per year [9, 17, 37]; conversely network availability [11, 40, 54] and network latencies improve slowly, and long latencies and failures waste human time.

Current operating systems and networks do not provide good support for aggressive background transfers. In particular, because background transfers compete with foreground requests, they can hurt overall performance and availability by increasing network congestion. Applications must therefore carefully balance the benefits of background transfers against the risk of both *self-interference*, where applications hurt their own performance, and *cross-interference*, where applications hurt other applications' performance. Often, applications attempt to achieve this balance by setting "magic numbers" (e.g., the prefetch threshold in prefetching algorithms [18, 26]) that have little obvious relationship to system goals (e.g., availability or latency) or constraints (e.g., current spare network bandwidth).

Our goal is for the operating system to manage network resources in order to provide a simple abstraction of zero-cost background transfers. A self-tuning background transport layer will enable new classes of applications by (1) simplifying applications, (2) reducing the risk of being too aggressive, and (3) making

*This work was supported in part by an NSF CISE grant (CDA-9624082), the Texas Advanced Technology Program, the Texas Advanced Research Program, and Tivoli. Dahlin was also supported by an NSF CAREER award (CCR-9733842) and an Alfred P. Sloan Research Fellowship.