Network Working Group
Request for Comments: 2140
Category: Informational

J. Touch
ISI

April 1997

TCP Control Block Interdependence

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This memo makes the case for interdependent TCP control blocks, where part of the TCP state is shared among similar concurrent connections, or across similar connection instances. TCP state includes a combination of parameters, such as connection state, current round-trip time estimates, congestion control information, and process information. This state is currently maintained on a per-connection basis in the TCP control block, but should be shared across connections to the same host. The goal is to improve transient transport performance, while maintaining backward-compatibility with existing implementations.

This document is a product of the LSAM project at ISI.

Introduction

TCP is a connection-oriented reliable transport protocol layered over IP [9]. Each TCP connection maintains state, usually in a data structure called the TCP Control Block (TCB). The TCB contains information about the connection state, its associated local process, and feedback parameters about the connection’s transmission properties. As originally specified and usually implemented, the TCB is maintained on a per-connection basis. This document discusses the implications of that decision, and argues for an alternate implementation that shares some of this state across similar connection instances and among similar simultaneous connections. The resulting implementation can have better transient performance, especially for numerous short-lived and simultaneous connections, as often used in the World-Wide Web [1]. These changes affect only the TCB initialization, and so have no effect on the long-term behavior of TCP after a connection has been established.

The TCP Control Block (TCB)

A TCB is associated with each connection, i.e., with each association of a pair of applications across the network. The TCB can be summarized as containing [9]:

- Local process state
    - pointers to send and receive buffers
    - pointers to retransmission queue and current segment
    - pointers to Internet Protocol (IP) PCB
- Per-connection shared state
    - macro-state
        - connection state
        - timers
        - flags
        - local and remote host numbers and ports
    - micro-state
        - send and receive window state (size*, current number)
        - round-trip time and variance
        - cong. window size*
        - cong. window size threshold*
        - max windows seen*
        - MSS#
        - round-trip time and variance#

The per-connection information is shown as split into macro-state and micro-state, terminology borrowed from [5]. Macro-state describes the finite state machine; we include the endpoint numbers and components (timers, flags) used to help maintain that state. This includes the protocol for establishing and maintaining shared state about the connection. Micro-state describes the protocol after a connection has been established, to maintain the reliability and congestion control of the data transferred in the connection.

We further distinguish two other classes of shared micro-state that are associated more with host-pairs than with application pairs. One class is clearly host-pair dependent (#, e.g., MSS, RTT), and the other is host-pair dependent in its aggregate (*, e.g., cong. window info., curr. window sizes).

TCB Interdependence

The observation that some TCB state is host-pair specific rather than application-pair dependent is not new, and is a common engineering decision in layered protocol implementations. A discussion of sharing RTT information among protocols layered over IP, including UDP and TCP, occurred in [8]. T/TCP uses caches to maintain TCB information across instances, e.g., smoothed RTT, RTT variance, congestion avoidance threshold, and MSS [3]. These values are in addition to connection counts used by T/TCP to accelerate data delivery prior to the full three-way handshake during an OPEN. The goal is to aggregate TCB components where they reflect one association - that of the host-pair, rather than artificially separating those components by connection.

At least one current T/TCP implementation saves the MSS and aggregates the RTT parameters across multiple connections, but omits caching the congestion window information [4], as originally specified in [2]. There may be other values that may be cached, such as current window size, to permit new connections full access to accumulated channel resources.

We observe that there are two cases of TCB interdependence. Temporal sharing occurs when the TCB of an earlier (now CLOSED) connection to a host is used to initialize some parameters of a new connection to that same host. Ensemble sharing occurs when a currently active connection to a host is used to initialize another (concurrent) connection to that host. T/TCP documents considered the temporal case; we consider both.

An Example of Temporal Sharing

Temporal sharing of cached TCB data has been implemented in the SunOS 4.1.3 T/TCP extensions [4] and the FreeBSD port of same [7]. As mentioned before, only the MSS and RTT parameters are cached, as originally specified in [2]. Later discussion of T/TCP suggested including congestion control parameters in this cache [3].

The cache is accessed in two ways: it is read to initialize new TCBs, and written when more current per-host state is available. New TCBs are initialized as follows; snd_cwnd reuse is not yet implemented, although discussed in the T/TCP concepts [2]:

TEMPORAL SHARING - TCB Initialization

Cached TCB	New TCB	
old-MSS	old-MSS	
old-RTT	old-RTT	
old-RTTvar	old-RTTvar	
old-snd_cwnd	old-snd_cwnd	(not yet impl.)

Most cached TCB values are updated when a connection closes. An exception is MSS, which is updated whenever the MSS option is received in a TCP header.

TEMPORAL SHARING - Cache Updates

Cached TCB	Current TCB	when?	New Cached TCB	
old-MSS	curr-MSS	MSSopt	curr-MSS	
old-RTT	curr-RTT	CLOSE	old += (curr - old) >> 2	
old-RTTvar	curr-RTTvar	CLOSE	old += (curr - old) >> 2	
old-snd_cwnd	curr-snd_cwnd	CLOSE	curr-snd_cwnd	(not yet impl.)

MSS caching is trivial; reported values are cached, and the most recent value is used. The cache is updated when the MSS option is received, so the cache always has the most recent MSS value from any connection. The cache is consulted only at connection establishment, and not otherwise updated, which means that MSS options do not affect current connections. The default MSS is never saved; only reported MSS values update the cache, so an explicit override is required to reduce the MSS.

RTT values are updated by a more complicated mechanism [3], [8]. Dynamic RTT estimation requires a sequence of RTT measurements, even though a single T/TCP transaction may not accumulate enough samples. As a result, the cached RTT (and its variance) is an average of its previous value with the contents of the currently active TCB for that host, when a TCB is closed. RTT values are updated only when a connection is closed. Further, the method for averaging the RTT values is not the same as the method for computing the RTT values within a connection, so that the cached value may not be appropriate.

For temporal sharing, the cache requires updating only when a connection closes, because the cached values will not yet be used to initialize a new TCB. For the ensemble sharing, this is not the case, as discussed below.

Other TCB variables may also be cached between sequential instances, such as the congestion control window information. Old cache values can be overwritten with the current TCB estimates, or a MAX or MIN function can be used to merge the results, depending on the optimism or pessimism of the reused values. For example, the congestion window can be reused if there are no concurrent connections.

An Example of Ensemble Sharing

Sharing cached TCB data across concurrent connections requires attention to the aggregate nature of some of the shared state. Although MSS and RTT values can be shared by copying, it may not be appropriate to copy congestion window information. At this point, we present only the MSS and RTT rules:

ENSEMBLE SHARING - TCB Initialization

Cached TCB	New TCB
old-MSS	old-MSS
old-RTT	old-RTT
old-RTTvar	old-RTTvar

ENSEMBLE SHARING - Cache Updates

Cached TCB	Current TCB	when?	New Cached TCB
old-MSS	curr-MSS	MSSopt	curr-MSS
old-RTT	curr-RTT	update	rtt_update(old,curr)
old-RTTvar	curr-RTTvar	update	rtt_update(old,curr)

For ensemble sharing, TCB information should be cached as early as possible, sometimes before a connection is closed. Otherwise, opening multiple concurrent connections may not result in TCB data sharing if no connection closes before others open. An optimistic solution would

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**