The Protocol Bypass Concept for High Speed OSI Data Transfer

C. M. Woodside*, K. Ravindran and R. G. Franks***
* Telecommunications Research Institute of Ontario, Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada K1S 5B6
** Dept. of Computer and Information Sciences, Kansas State University, Manhattan KS 66506

Abstract

A protocol bypass is a fast processing path which is used for some data units, for instance for large data packets. It can provide conformance with standardized layered protocol system like OSI, together with some of the performance benefits of a "lightweight" protocol. The concept is discussed as it applies to the movement of user data by the OSI transport and session protocols, with some implementation experience, and an outline for an approach to formally deriving bypass specifications from protocol specifications is given. This outline uses some steps which must still be proven to be practical. Correct interleaving of data units from the two paths is a major concern, especially with multiple asynchronously specified layers. It seems that the difficulty can be overcome and the concept has promise. In the (rather limited) implementation, bypassing consistently outperformed parallel processing as a means of performance enhancement.

1 Introduction

The search for generality, flexibility and standardization in communications protocols has led to the OSI layered system [1],[2] and many offshoots such as MAP [3]. However, the slowness of execution of the protocol implementations, which is essentially due to the complex checking of conditions that is done at every operation, is becoming a limiting factor in some applications. Therefore a new generation of lightweight or high-speed data transfer protocols is now emerging.

The lightweight protocols exploit the low error rates of many networks, and use larger packets, reduced options to unclutter the data path, and more efficient methods for congestion control. They are partly motivated by the high speeds of new fiber-based networks, partly by high-throughput applications such as file system backups and full motion video, and partly by performance constraints already being felt with current protocols. Examples are Zwaenepoel's Blast protocol [4, 5], VMTP [6], and XTP [7].

Present-day "heavyweight" protocols such as OSI place notable performance constraints on distributed applications. Svobodova [8] surveys the status of transport protocols (OSI and others) running on LANs and finds throughputs up to a maximum of about 2 Mbits/s (other references are found in her paper). The goal of the current lightweight

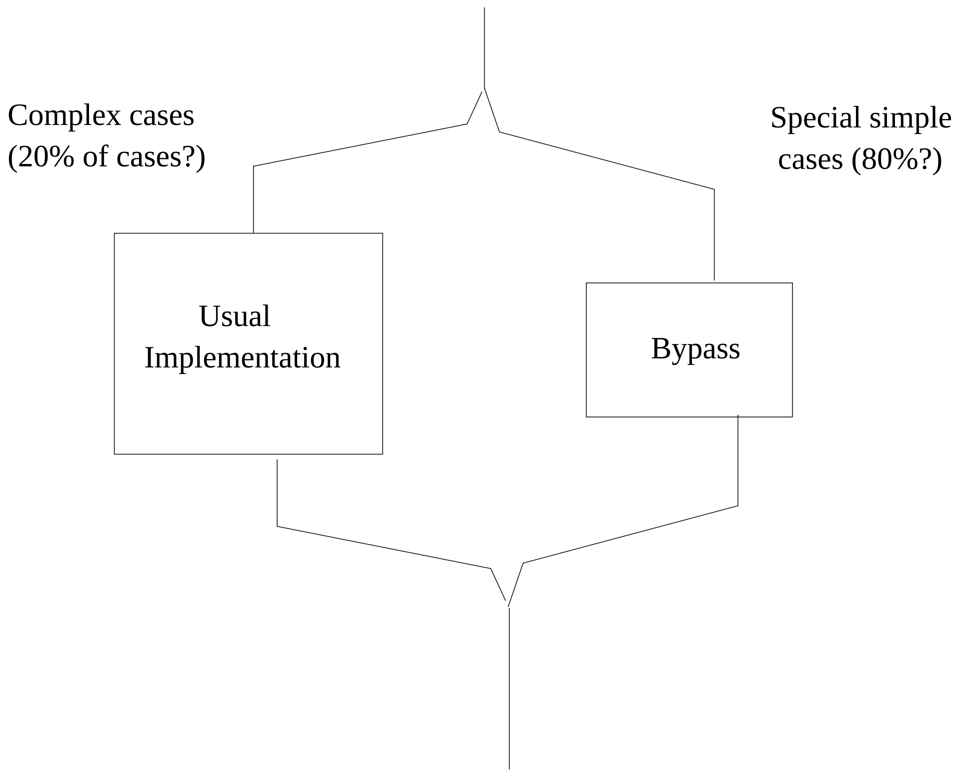


Figure 1: The Bypass Concept

protocols is to obtain an improvement by an order of magnitude now, and much greater increases later on faster processors and nets, into the range 100–1000 Mbits/s. The protocols do not have to be completely new ones; Clark et al [9] have analyzed the fundamental limitations in IP/TCP to show that it could be operated in the 100 Mbit/s. range if implemented properly. They describe a "fast path" concept in [9],[10] which has already, in an existing workstation environment, moved files at close to the capacity of an Ethernet. Jacobson has called his guiding principle for a fast receiving path "Header Prediction".

The bypass concept described here is an attempt to generalize Jacobson's "Header Prediction" to other protocols, particularly to layered protocol systems like OSI. Figure 1 shows the general notion of The bypass part is activated when a very limited set of conditions apply which may be executed quickly; it will be effective if the conditions are satisfied often enough. Hopefully, one can obtain both the performance benefits of a lightweight protocol, when operating within a restricted set of circumstances, and the standardization and functional benefits of a fully conforming standard protocol stack. To obtain the full potential performance, some adaptation of the OSI standards may be necessary (e.g., larger window sizes). The bypass is an implementation concept rather than a new protocol, but it raises interesting questions about specification, such as

- specifying the bypass path and control when several protocol layers are involved,
- specification properties which could constrain the applicability of the concept in

some cases,

- formal derivation of the bypass process specification from the protocol specification.

The bypass concept is based on a standard idea in performance enhancement, called the 'centering principle' by Smith [11] and sometimes called the "80–20 principle". In 'centering', special fast processing is provided for a frequently-occurring case. Although the idea is standard, and seems to be relatively straightforward in [11], it is nontrivial for layered protocols because of the sequential relationships generated by segmenting and reassembly of data, the complex interchanges supported by each layer, the web of relationships that may exist between entities at different layers due to multiplexing of connections, and especially because of the asynchronous nature of the separate layers.

The purpose of this paper then is to explore the feasibility of constructing a bypass for an arbitrary layered protocol, and more particularly for OSI. It begins with an informal outline of a particular bypass for the OSI Session and Transport layers, identifying difficulties associated with operating and controlling a bypass, that seem to be fundamental. Ways to overcome these difficulties are discussed, and have been demonstrated by some preliminary implementation experience discussed in Section 6. The formal derivation of bypass specifications is approached in Section 7 by outlining how the simple control strategy used in Sections 3, 4 and 5 could be described formally. The paper is intended to show that bypassing is a fruitful and important mechanism for combining standard and lightweight protocol concepts, and that this topic is worth pursuing further.

2 The Architecture of a Bypass

Figure 2(a) illustrates the architecture of a bypass, including the following nomenclature:

stack the set of entities implementing the standard protocol, which are to be bypassed,

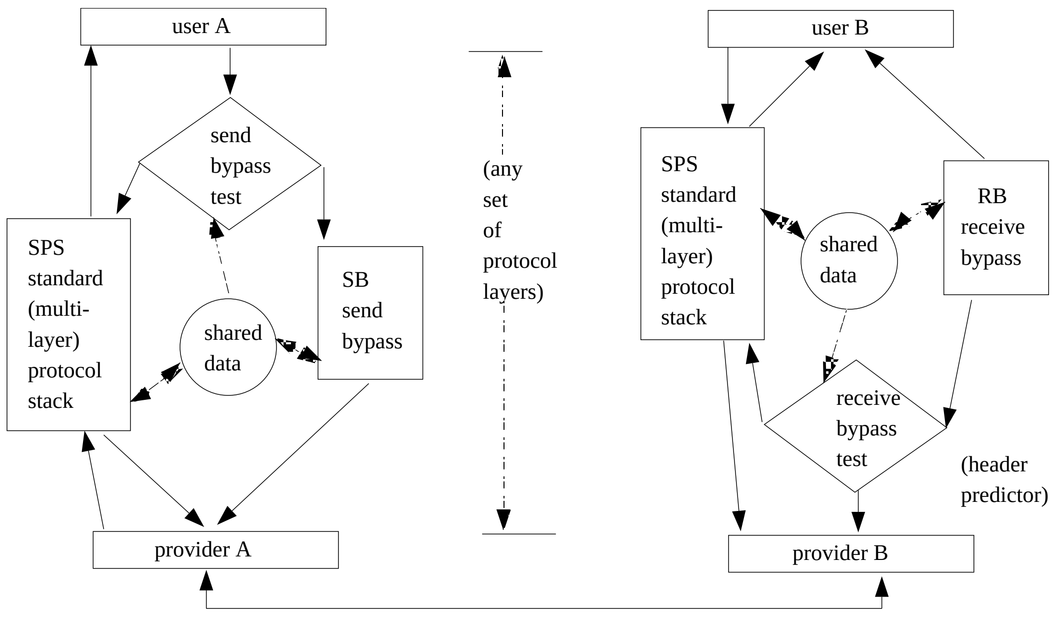
user the user of the topmost layer of 'stack', assumed to be unique,

provider the service provider entity below the bottom layer of 'stack', also assumed to be unique at each end. These embody a service layer serving both ends, and hiding the physical transmission of data, etc.

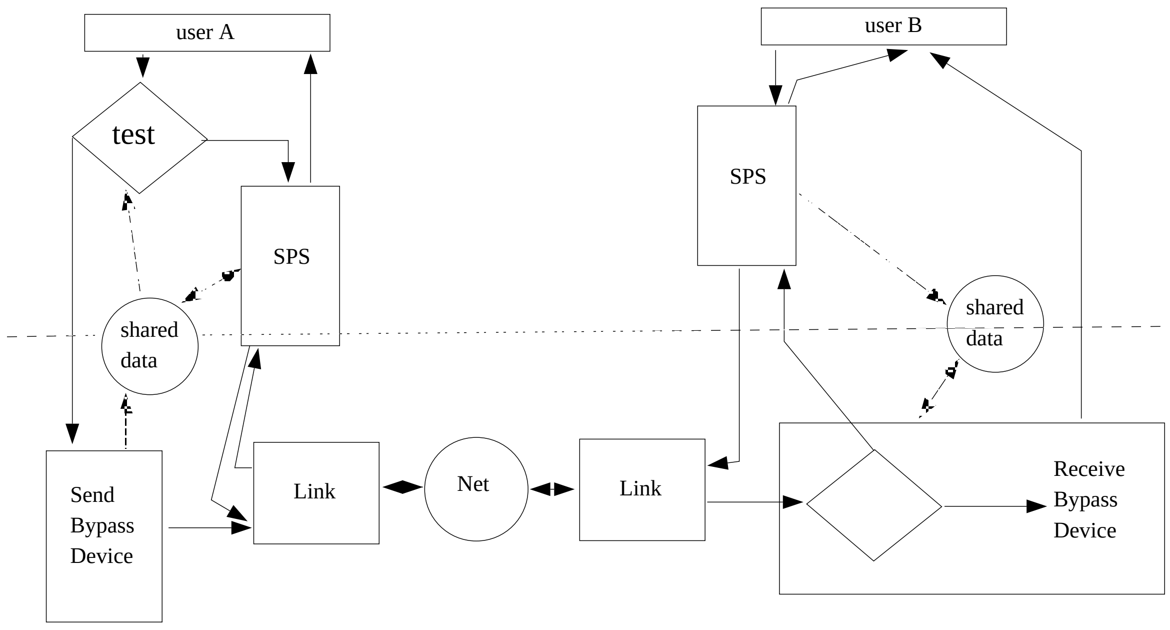
The Figure shows a system to support one-way data transfer, in which data flows from the 'Data Sender' end A to the 'Data Receiver' end B; note that control packets are also involved and flow in both directions.

As a data unit flows from left to right, it is filtered by tests and if it qualifies, is processed by the fast path. The fast path is envisaged as a single process without internal concurrency, cutting through all the layers of 'stack', while 'stack' may have internal concurrent processes. Jacobson's 'Header Prediction' conforms somewhat to this model, for in his system a received packet is examined for a match against a stored predicted header at the test labelled RBYPASSTEST, but there is no SEND test.

One of the potential values of isolating the fast path rather than just optimizing the code in 'stack' is the excellent prospect of implementing the bypass in hardware. Figure 2(b) shows a suggested division when the service entities are hardware-implemented link controllers such as Ethernet controller chips. SBP and RBP are the 'send' and 'receive' bypass devices. The receive test RBYPASSTEST is in hardware, to avoid breaking the hardware path to the user level, and the RBP device writes data into addresses supplied in memory accessed by the device. The send test does not have such compelling reasons to be in hardware, and so is not.



(a) Architecture of Bypass for Sender and Receiver



(b) A Hardware Implementation Strategy with two bypass devices (for the case where the provider is the link controller)

Figure 2: Bypass Architectures

3 One-Way Bypass of OSI Session and Transport Layers

To focus the discussion this section examines a bypass around an OSI session layer combined with a Transport Class 2 layer, and used with extended TPDU-numbering, but without flow control.

3.1 Ideal Data Transfer

Further, bypassing will be provided only for those data units and transfers which satisfy the following conditions for "ideal data transfer",

- the session service data units are of type "data",
- there is no segmentation/reassembly, concatenation/separation, or multiplexing in the two layers,
- there are no piggybacked acknowledgements or window credits

Attention is also restricted to a *one-way* bypass, as illustrated in Figure 2(a). It could be used for instance in a file server with a send-only bypass loading pages across a network to diskless workstations with receive-only bypasses. Only one-way "fast paths" were considered in [9, 10]. Greater generality will be introduced in later sections.

In an "ideal data transfer" each SSDU corresponds to exactly one NSDU and entails these operations in the layers:

- increment PDU numbers by one in each layer at sender and receiver,
- send an acknowledgement, after a receive.
  The bypass must produce the same effects as a standard implementation.

  The general architecture of a one-way bypass system is as shown in Figure 2(a).

3.2 The Bypass Tests, without Window Flow Control

Correct execution of the protocol by a bypass depends on two key operations – the identification, at the bypass-condition tests in Figure 2(a), of those data units which which will give an ideal transfer, and correct interleaving of data units which follow the two paths. On sending a user data unit, the conditions to be met are that it be small enough to fit into one TPDU, and that it be a 'data' type of unit (rather than control). The connection must also meet the conditions stated above (eg., no multiplexing). Therefore the bypass test will be only partly based on headers (as it was in Jacobson's method); it will also include some aspects of the state of the protocol layers, shown as the "shared data" in Figure 2(a).

Packets which fail the bypass test follow the normal processing path through the full implementation. This raises the problem of ensuring correct interleaving of data units from the two paths where they rejoin. This is a non-trivial problem because in principle the separate layers in OSI are asynchronous entities. In this work interleaving is ensured by further assuming or enforcing that the protocol execution for an ideal data transfer is atomic with respect to all other operations (send or receive, and by either layer) for the same connection. This extra assumption is called 'relative atomicity' of processing. Then there is a simple toggle in the flow, switching from one path to the other, and the

DOCKET ALARM

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

fastcase® Smarter legal research.