SLIPSTREAM: AUTOMATIC INTERPROCESS COMMUNICATION OPTIMIZATION

Will Dietz, Joshua Cranmer, Nathan Dautenhahn, Vikram Adve
Introduction

- Use of TCP is ubiquitous
  - Widely Supported
  - Location Transparency
- Programmer-friendly but not always ideal
- Faster IPC exists for local communication
- Goal: Best of TCP generality with fast performance locally
Motivating Example

Web App (Java, C, Python, …)  Memory Store  Database

TCP/IP  Kernel

Clients

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Motivating Example

Optimization opportunity: Local communication could use UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.
Motivating Example – Manually Optimized

Kernel

Clients

Web App
(Java, C, Python, …)

Memory Store

Database

TCP/IP

UDS vs TCP: up to 2x ops/sec!

Avoid TCP layers

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Motivating Example – Manually Optimized

Can we do this automatically?

Slipstream: Automatic IPC Optimization

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
What’s Coming

- Slipstream Overview
- How Slipstream Works
- Selected Results
  - Performance
  - Compatibility
  - Docker
- Conclusions
Slipstream Overview
Design Principles: Automatic and Useful

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

Diagram:
- Web App (Java, C, Python, …)
- Memory Store
- TCP/IP
- Clients
- No App. Modifications
- Language Agnostic
- No Kernel Modifications
- No Network Assumptions
- Backwards Compatible

http://wdtz.org/slipstream
Slipstream Architecture: What is it?

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
Example + (libipc)

libipc: Shim library between application and OS

Web App (Java, C, Python, …)
Memory Store
Database

TCP/IP
Kernel

Clients
Example + (libipc + ipcd)

Web App (Java, C, Python, ...)

Memory Store

Database

TCP/IP

Kernel

Clients

libipc

ipcd

libipc

ipcd

libipc

ipcd

ipcd:
Coordinating daemon for all libipc on host

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.
Example + Slipstream

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
How Slipstream Works
How Slipstream Works: Overview

1. Track TCP State
2. Detect Local Communication
3. Switch to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Tracking TCP State

Kernel

Clients

Web App (Java, C, Python, …)

Memory Store

Database

TCP/IP

libipc

ipcd

libipc

libipc

ipcd

 KERNEL

Track TCP State

Detect Local Communication

Switch to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
1) Tracking TCP State

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

Track TCP State

Detect Local Communication

Switch to UDS

libipcd: Transparently monitors TCP operations

Web App (Java, C, Python, …)
Memory Store
Database

TCP/IP

Kernel

TCP/IP

Kernel

read
write
listen

…
1) Tracking TCP State

- Present by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

- **libipc**: Maintains state throughout execution
- **libipc**: Transparently monitors TCP operations

Track TCP State -> Detect Local Communication -> Switch to UDS

Clients

- Web App (Java, C, Python, …)
- Memory Store
- Database

TCP/IP

Libipc:

- `fork`
- `exec`
- `dup`

Read

Write

Listen
Detecting Local Communication

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Detecting Local Communication

Web App (Java, C, Python, …) → libipc → ipcd → libipc

Kernel

TCP/IP

ipcd:
Uses information from libipc to perform endpoint matching

Track TCP State ➔ Detect Local Communication ➔ Switch to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Detecting Local Communication

Endpoint Matching:
1. \{SRC IP:Port, DEST IP:Port\}
2. N-Byte Checksum of Stream Data
3. Timing Windows

Track TCP State
Detect Local Communication
Switch to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
2) Detecting Local Communication

Web App (Java, C, Python, …) → Memory Store → Database

Libipc → Ipcd → Libipc

Local communication detected!

Track TCP State → Detect Local Communication → Switch to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Switching to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Switching to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

ipcd:
Generate optimized socketpair, send to libipc instances

Track TCP State
Detect Local Communication
Switch to UDS
Switching to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
Switching to UDS

Web App (Java, C, Python, …) → Memory Store

libipc: Preserve expected socket behavior for compatibility

Track TCP State → Detect Local Communication → Switch to UDS

libipc: Transparently switch to new transport

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Switching to UDS

Web App (Java, C, Python, …) -> Memory Store

Kernel

TCP/IP

Slipstream: up to 2x ops/sec

Detect Local Communication

Switch to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Selected Results

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.
Performance vs Local TCP

Memcached

Up to +80% ops/sec!

At least +40% ops/sec!
Performance vs Manual UDS

Netperf UDS

Slipstream automatically achieves performance similar to manual UDS!
Performance Overhead

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

Netperf

~3.5% overhead

Memcached

< 3% overhead
Software Compatibility

- **Supports many popular server applications**
  - MySQL, PostgreSQL, Redis, Memcached, Apache, Jenkins, …

- **Language agnostic** in practice
  - Java, C, Python, …

- **All work correctly with Slipstream**
- **All are optimized when communicating locally**

- **Supports use with Docker!**
Example with Docker

[Diagram showing components and their connections]

- Web App (Java, C, Python, …)
- Memory Store
- Database
- TCP/IP
- Kernel
- Clients

Put app in container
Container per component
Slower comm., extra layers

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.
Example with Docker + Slipstream?

Did someone say they need Automatic IPC Optimization?

- Web App (Java, C, Python, …)
- Memory Store
- Database

- Put app in container
- Container per component
- Slower comm., extra layers

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
Slipstream with Docker

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Slipstream with Docker

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.
Slipstream with Docker

Kernel

Clients

Web App
(Java, C, Python, …)

Memory Store

TCP/IP

ipcd

libipc

Database

ipcd can run in a container too

Connect to common ipcd

libipc per container

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Slipstream with Docker

Optimize across containers!

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Slipstream with Docker

- Web App (Java, C, Python, ...)
- Memory Store
- Database

Optimize across containers!

2x-3x ops/sec over Docker

Same perf. as without Docker

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Docker with NetPIPE-C

Slipstream shows **large speedups**

**Same perf.** with or without Docker!

Baseline Docker is significantly slower

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Conclusions:

- **Automatically** use faster IPC mechanism when available
- **Easy to deploy, highly compatible**
  - No OS modifications
  - No modifications to application
- **Language-agnostic**
- **Backwards-compatible** (partial deployment supported)
- ~2x bandwidth improvement for host-local communication
- **Low overhead** when optimization is not possible
- **Freely Available** at [http://wdtz.org/slipstream](http://wdtz.org/slipstream)

Thank you! Questions?
(End of presentation)
Q: Why use UDS instead of other local IPC?

- Simply: Socket interface of UDS facilitated prototyping
- Any local IPC mechanism should be possible
  - (Patches welcome! 😊)
- Others could **avoid kernel interaction** for even better performance
  - See ipc-bench paper
- **Slipstream is not UDS-specific**
  - UDS implementation detail, idea is faster performance leveraging locality
  - In fact, netperf sometimes slower with UDS (see paper).
Q: Why not have application do this itself?

- **Legacy**: many applications don’t do this today
  - (Knowledge that UDS faster than TCP is not new!)
- **They shouldn’t have to**, Slipstream! 😊
  - Achieves performance of manual optimization
  - Low overhead when optimization not possible
- **Applications have better things to do than transport selection!**
Q: Why userspace solution, not in-kernel?

- Mostly: **Legacy**

- Ease of use, ease of deployment

- Suitable for use on other systems (FreeBSD, etc.)

- May not be appropriate for kernel
  - Correctness concerns
    - (Opt-in approach likely appropriate)

- Possibility for using even more efficient transports than UDS
  - Can’t cut out kernel with a kernel-based solution
Q: What about frequent short-lived connections?

- Current implementation not well-suited for this, but could be!
- Would not benefit from optimization
  - only applied to longer-lived connections
- In fact, initial connection latency over localhost increased
  - Not by design, result of implementation choices (but nevertheless)
- Probably better off with custom “is local” decision mechanism
  - Ex: “Is destination IP 127.0.0.1”
  - Would allow optimizing before connection, get faster connect() times
- Thanks for the question, Joshua 😊.
## Related Work

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Application Transparency</th>
<th>OS Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS Feature</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows Fast-Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Solaris TCP Fusion</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AIX fastlo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Linux TCP Friends</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td><strong>VM-VM</strong></td>
<td>XWay</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XenSocket</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XenLoop</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Userspace Networking Stack</strong></td>
<td>mTCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sandstorm</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Userspace Shim Library</strong></td>
<td>Fable</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Java Fast Sockets</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Universal Fast Sockets</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>FastSockets</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Slipstream</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC '15. [http://wdtz.org/slipstream](http://wdtz.org/slipstream)
Building on Slipstream

- Alternative transports (shared memory, etc.) (see ipc-bench)
  - Select best transport for workload/configuration

- Support multiple locality detection mechanisms
  - Provided one is suitable for many use-cases
  - If can assume network topology, can directly determine this

- More exhaustive testing, performance investigation
Try it out today!

- No configuration, simple deployment
- Big performance gains for many applications (Docker support!)
- Low overheads when optimization is not possible
- Makes most of partial deployment
- Open-source, CRAPL (don’t judge, it’s research!)
- Download today!

[http://wdtz.org/slipstream](http://wdtz.org/slipstream)
Deploying Slipstream
Example + Slipstream: Deployment
Example + Slipstream: Deployment

libipc deployed with application
(LD_PRELOAD or /etc/ld.so.preload)
Challenges and Solutions
Example + Slipstream: Key Challenge

Challenge:
Which connections are communicating locally?

Solution:
Endpoint matching algorithm to identify local communication

libipc only sees what goes through it

Kernel
Web App (Java, C, Python, …)
Memory Store
Database

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
http://wdtz.org/slipstream
Endpoint Matching Overview

It’s the orange one!

libipc works with ipcd which has more information

Observed behavior from both libipc instances used by ipcd to find match
Endpoint Matching Overview

libipc works with ipcd which has more information

What information can be used to help pair endpoints?

libipc

ipcd

behavior from both libipc used by ipcd to find match

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Endpoint Matching Overview

libipc works with ipcd which has more information.

Observed behavior from both libipc instances used by ipcd to find match.
Endpoint Matching Overview

libipc works with ipcd which has more information

Observed behavior from both libipc instances used by ipcd to find match

libipc works with ipcd which has more information

Observed behavior from both libipc instances used by ipcd to find match

Src/Dst IP and Port

Data Sent/Recv

Timing Details

More details in paper!
Endpoint Matching Overview

libipc works with ipcd which has more information

Observed behavior from both libipc instances used by ipcd to find match
Endpoint Matching Overview

libipc works with ipcd which has more information

Observed behavior from both libipc instances used by ipcd to find match

Local endpoints found!
Endpoint Matching: Complete

Local endpoints found!

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Automatic IPC Optimization

Kernel

Clients

Web App
Java, C, Python, ...

Memory Store

Database

TCP/IP

libipc

ipcd

libipc

ipcd

libipc

ipcd

Transparently switches to UDS

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
Automatic IPC Optimization

UDS vs TCP: up to 2x ops/sec
Without Modifying Applications?

Implementation Challenge:

libipc must preserve expected socket behavior for compatibility

Solution:
Engineering!

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
Results, with slides.

Faster than TCP? NetPIPE Microbenchmarks

- NetPIPE-C
- NetPIPE-Java

1.4x to 2.5x improvement!

Works with both C and Java!
Faster than TCP? Memcached

1.4x to 1.8x ops/sec
Faster than TCP? PostgreSQL

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

PostgreSQL – TPC-B

PostgreSQL - Select

No improvement when CPU/IO bound

Small improvement for Select-only

http://wdtz.org/slipstream
Faster than TCP? PostgreSQL

Application benefit depends on how much network is the bottleneck

No improvement when CPU/IO bound

Small improvement for Select-only

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.
Slower than TCP? Slipstream Overhead

**Netperf**

- ~3.5% overhead for Netperf

**Memcached**

- <3% overhead for Memcached

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Slower than TCP? Slipstream Overhead

Overhead incurred is low in even network-intensive applications

~3.5% overhead for Netperf

< 3% overhead for Memcached

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Slower than manual UDS? Netperf Microbenchmark

Relative to fixed benchmark

(Default UDS does poorly)
Slower than manual UDS? Netperf Microbenchmark

Automatically achieves performance close to manually using UDS!

(Default UDS does poorly)
Docker with NetPIPE-C

Slipstream shows **large speedups**

Same perf. with or without Docker!

Baseline Docker is significantly slower

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Docker with Memcached

Memcached with Docker

Curious: Memcached with Slipstream performs better across Docker containers

Baseline Docker has lower performance

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream
Software Compatibility

- **Supports many popular server applications**
  - MySQL, PostgreSQL, Redis, Memcached, Apache, Jenkins, ...

- **Language agnostic** in practice
  - Java, C, Python, ...

- And of course our microbenchmarks:
  - NetPIPE, netperf, iperf, Imbench

- All work correctly with Slipstream
- All are optimized when communicating locally

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC’15.

http://wdtz.org/slipstream
Slipstream Overview – Multiple Applications

- **ipcd**: One instance per host
- **libipc**: One instance per process

Presented by Will Dietz, University of Illinois at Urbana-Champaign. USENIX ATC'15.

http://wdtz.org/slipstream