# Efficient UDP Haptic Device Communication

Daniel Wilhelm
***Advisor**: Prof. Hong Tan, **Graduate Student**: Ryan Traylor*
*Dr. Dov Adelstein (NASA Ames)*

Haptic Interface Research Laboratory
Undergraduate Research Assistant
(dwilhelm@purdue.edu)

*SURI Symposium 2004. August 6, 2004.*

## Haptic Device Constraints

- Human Eye: 30 Hz (33 ms/frame)
- Human Skin: 1000 Hz (1 ms/frame)

What must meet this constraint?

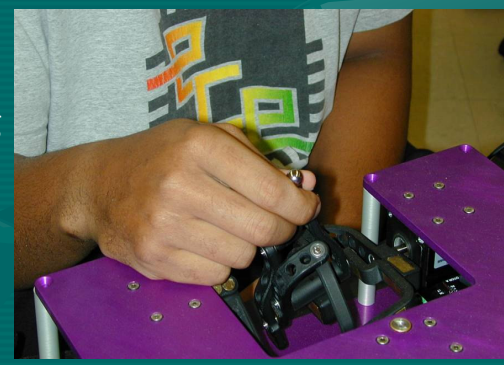
- Computer
- Haptic Device
- Network

2

# MiniStick

***In-production HIRL haptic device:***

- High-precision motors and encoders
- 3 Degrees of freedom
- Passive mass balancing
- LAN Ethernet control



3

# UDP Advantages

***User Datagram Protocol Advantages:***

- Small overhead
- Control from any computer(s) on LAN
- Broadcast to many computers without slowdown
- Send and forget

4

# UDP Disadvantages

***User Datagram Protocol Disadvantages:***

- Reception not guaranteed (dropped datagrams)
- Order not guaranteed (out-of-order datagrams)
- Duplicate datagrams

5

# UDP Transmission Speed

***Speed/Efficiency must be maximized in order to:***

- Ensure smooth haptic perception
- Allow transmission over congested networks
- Enable time-intensive calculations/graphics

6

# Round Trip Frequency (RTF)

1. Host computer sends a 64-byte UDP packet to the MiniStick.
2. MiniStick receives the packet, processes it, then sends a 64-byte UDP reply.
3. The host computer receives the reply.

7

# Theoretical RTF

$$\frac{2\,packets}{1}\cdot\frac{64bytes}{packet}\cdot\frac{8bits}{byte}\cdot\frac{s}{10^6\,bits}\approx 130\mu s$$

Internally, the MiniStick takes approximately the same amount of time to process a packet. Thus,

$$\frac{1}{130\mu s+130\mu s}\approx 3800Hz$$

8

## Software Development

***Software was developed:***

- Precise timing mechanisms (μs)
- Averaged across groups of packets
- Automated packet validity testing
- Automated testing

9

## Software Development

***Initial Software versions:***

- Linux version (ran on Knoppix and Red Hat)
- Serial I/O
- Overlapped I/O
- Send/Receive multi-threading

10

# Initial Empirical RTF

**LabView:** ~3200 Hz then stabilizes at ~1800 Hz

**C:** ~3800 Hz then stabilizes at ~2000 Hz

- **Why the drop in RTF? Is the higher RTF desirable?**
- LabView is slower due to extraneous debugging code.

11

# Networking Modes

1. **Real-Time –** 2000 Hz
2. **Buffered –** 3800 Hz
3. **Distributed –** theoretical maximum

12

## Real Time (2000 Hz)

Receives data slowly after waiting for it.

Note that receive times are ***more than*** the predicted 230 µs.

| | |
|---|---|
| 7E-05 | send |
| 0.000431 | receive |
| 7.3E-05 | send |
| 0.000431 | receive |
| 7.1E-05 | send |
| 0.000433 | receive |

(in seconds)

13

## Buffered (3800 Hz)

Receives last-frame data quickly from buffer.

Note that receive times are ***less than*** the predicted 230 µs.

| | |
|---|---|
| 7E-05 | send |
| 0.00019 | receive |
| 7E-05 | send |
| 0.000191 | receive |
| 7E-05 | send |
| 0.000193 | receive |

(in seconds)

14

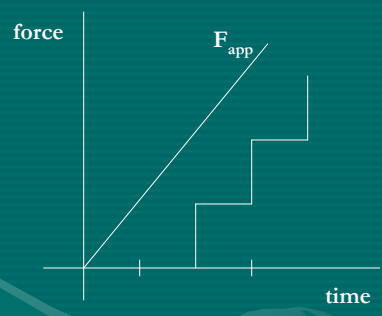
# Distributed (Max Hz)

Multiple computers used to alternately send packets.

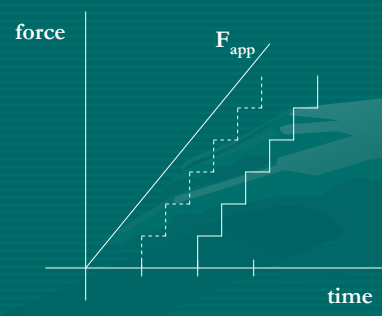
The idea is to *force* the MiniStick to be the bottleneck – the past two methods indicate the computer currently is.

15

# Distributed Dilemma




2000 Hz, Non-Distributed




4000 Hz, Distributed

**How does this affect stability and perceptual smoothness?**

16

## Possible Variables

- Internet vs. Direct-Connect
- Windows vs. Linux vs. RTLinux
- Berkeley Sockets vs. Windows Sockets

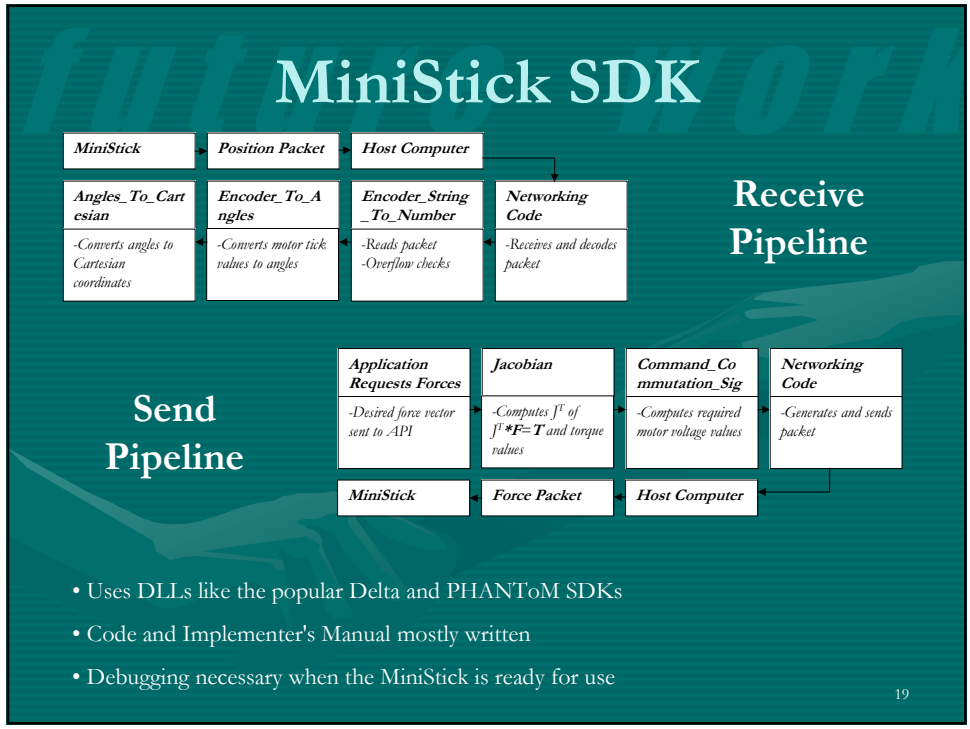
- Firewall?
- Buffered?
- Blocking?
- High/Low Priority Process?

17

## Empirical Results

| Experiment | Average Hz |
|---|---|
| Buffered | 3800 |
| Firewall | 2025 |
| High Priority | 2099 |
| Low Priority | 2074 |

18


MiniStick SDK
MiniStick
Position Packet
Host Computer
Angles_To_Cartesian
-Converts angles to Cartesian coordinates
Encoder_To_Angles
-Converts motor tick values to angles
Encoder_String_To_Number
-Reads packet
-Overflow checks
Networking Code
-Receives and decodes packet
Receive Pipeline
Send Pipeline
Application Requests Forces
-Desired force vector sent to API
Jacobian
-Computes $J^T$ of $J^T*F=T$ and torque values
Command_Commutation_Sig
-Computes required motor voltage values
Networking Code
-Generates and sends packet
MiniStick
Force Packet
Host Computer
• Uses DLLs like the popular Delta and PHANToM SDKs
• Code and Implementer's Manual mostly written
• Debugging necessary when the MiniStick is ready for use
19