aibo with ROS

Tomoya Fujita

R&D Center
Sony Corporation

R&D Center  System Technology Development Division  Base System Development Department
Copyright 2018 Sony Corporation
What is “aibo”?
Coming up in US
Very soon
 Comes close to you &
 Makes interaction
Intelligent Processing (Cloud)

Edge Computing System

Intelligent Processing (Edge)

Situational Awareness

Behavior Planning

Recognition

Behavior Control

Sensing

Hardware

Mechatronics

User/Environment
Feeling Area
aibo can feel someone is there.

Attraction Area
aibo tries to get attention

Communication Area
aibo communicates with user/owner

Feeling Area: 5m
Attraction Area: 3m
Communication Area: 1.5m
<table>
<thead>
<tr>
<th>Sensor</th>
<th>Location</th>
<th>Perpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image (fisheye camera)</td>
<td>Front Face</td>
<td>Face / Body / Hand / Object / Color / QR code</td>
</tr>
<tr>
<td>Image (fisheye camera)</td>
<td>Back</td>
<td>SLAM(Simultaneous Localization and Mapping)</td>
</tr>
<tr>
<td>ToF (Time of Flight)</td>
<td>Front Face</td>
<td>Object Detection</td>
</tr>
<tr>
<td>PSD (Position Sensitive Detector)</td>
<td>Chest</td>
<td>Cliff Detection</td>
</tr>
<tr>
<td>Touch Pressure</td>
<td>Back</td>
<td>Patting / Slapping Detection</td>
</tr>
<tr>
<td>Touch Capacitive</td>
<td>Head, Chin</td>
<td>Patting Detection</td>
</tr>
<tr>
<td>6 axis (gyro x accelerometer)</td>
<td>Head</td>
<td>Lift / Cliff Detection</td>
</tr>
<tr>
<td>6 axis (gyro x accelerometer)</td>
<td>Body</td>
<td>Lift Detection Walk Balance Calibration</td>
</tr>
<tr>
<td>Human Detect</td>
<td>Body</td>
<td>Human Detection</td>
</tr>
<tr>
<td>Ambient Light</td>
<td>Body</td>
<td>Environment Prediction</td>
</tr>
<tr>
<td>Ground</td>
<td>Paw</td>
<td>Floor Contact Detection</td>
</tr>
</tbody>
</table>
Mic Holes
Happy birthday

Tell aibo, “Happy birthday!” or “Happy birthday to you!” and your aibo will sing a birthday song. Seat the birthday boy or girl in front of your aibo, and have aibo sing as a special present. Make birthdays magical with aibo!

Let aibo know how proud you are of him or her!
Creator: Sony Corporation

Version: 1.0.0

Copyrights: Copyright 2018 Sony Corporation
ROS Embedded Optimization
ROS Transport Overview

XMLRPC: configuration information
TCPROS/UDPROS: data payload such as topic and services.
ROS Transport Overview

- What’s the problem?
  - CPU resource stress
  - Latency and Throughput.

- Countermeasure Preconditions
  - No change required for applications.
  - Good affinity with TCP/UDP
  - small latency and high throughput.

Computation Robot

```
3045SBOTQPSU0WFSWJFX
8IBU'TUIFQSPCMFN
• $16SFTPVSDFTUSFTT
• -BUFODZBOE5ISPVHIQVU
$PVOUFSNFBTVSF1SFDPOEJUJPOT
• /PDIBOHFSFRVJSFEGPSBQQMJDBUJPOT
• (PPEBGGJOJUZXJUI5$16%1
• TNBMMMBUFODZBOEIJHIUISPVHIQVU

SPTNBTUFS
```

Our Scope

```
Extend UDSROS layer with TCPROS/UDPROS, but nothing else will be changed.
```

Display Node

```
- BQUPQ
%JTQMBZ
/PEF
9.-31$ 0VS4DPQF
&YUFOE6%4304MBZFSXJUI5$13046%1304
CVUOPUIJOHFMTFXJMMCFDIBOHFE
```
Expected Improvement as System

- **Source Code**
  - [https://github.com/tomoyafujita/ipc-bench](https://github.com/tomoyafujita/ipc-bench)

- **Environment**
  - skylake(amd64) / hikey(arm64)
  - CPU frequency governor is “performance”
  - 100byte data payload / 10000 test loop average

- **Performance Improvement for System**

<table>
<thead>
<tr>
<th>Category</th>
<th>Skylake</th>
<th>Hikey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency [usec]</td>
<td>UDS(STREAM)</td>
<td>1.82 us</td>
</tr>
<tr>
<td></td>
<td>UDS(DGRAM)</td>
<td>2.14 us</td>
</tr>
<tr>
<td></td>
<td>TCP/IP</td>
<td>3.14 us</td>
</tr>
<tr>
<td></td>
<td>UDP/IP</td>
<td>2.78 us</td>
</tr>
<tr>
<td>Through-Put</td>
<td>UDS(STREAM)</td>
<td>1.95 Gbps</td>
</tr>
<tr>
<td></td>
<td>UDS(DGRAM)</td>
<td>1.98 Gbps</td>
</tr>
<tr>
<td></td>
<td>TCP/IP</td>
<td>0.43 Gbps</td>
</tr>
</tbody>
</table>
Extended Transport Overview

System-A(OS instance)

- rosmaster
- Node (cpp)
- Node (py)
- roscpp
- rospy
- UDSROS
- TCPROS/UDPROS
- Network

System-B(OS instance)

- Node (cpp)
- Node (py)
- roscpp
- rospy
- UDSROS
- TCPROS/UDPROS

---

System Internal

System Boundary Aware, so it's automatically which connection should be used.
Improvement Result

- **HelloWorld Benchmark**
  - Publisher: Subscriber = 1:1
  - Synchronous test loop using shared memory.
  - Latency is the time window between publish and callback entrance.
  - Environment: amd64 (skylake) and arm64 (hikey)

- **Result Comparison**

<table>
<thead>
<tr>
<th>Platform</th>
<th>ROS Transport</th>
<th>Duration [nsec]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
</tr>
</tbody>
</table>
| Skylake  | TCP/IP | 46,582 | 287,173 | 70,393  
|          | UDS  | 42,877 | 286,130 | 64,796  |
| Hikey    | TCP/IP | 552,224 | 6,040,312 | 1,145,881 |
|          | UDS  | 435,833 | 10,075,833 | 872,981 |
Github Repository
https://github.com/fujitatomoya/ros_comm/tree/topic-kinetic-devel-uds-support

Pull Request
https://github.com/ros/ros_comm/pull/1510

Tomoya Fujita <Tomoya.Fujita@sony.com>
Barry Xu <Barry.Xu@sony.com>
Direct I/O with rosbag

- Reference
  - [https://github.com/osrf/robag_direct_write](https://github.com/osrf/robag_direct_write)

Data in userland will be stored directly to the storage device without CPU workload.

(*) memory alignment constraints
Epoll

**What is the problem?**
- The more ROS topic connection, the more CPU stress with `ros::spin()`
  - Waiting for the stimulus during `ros::spin` without any message transmission.
- Poll system call is not good enough to take care of many file descriptors.

**Countermeasure**
- Using `epoll` instead, to reduce CPU stress.
- Already introduces as following commit, so backport the fix into 1.12.7 `ros_comm`.

```diff
commit 9c0db37d9231003a7f162857e4aeb45675839609
Author: Mike Purvis <mike@uwmikey.com>
Date:  Thu Dec 21 11:31:08 2017 -0500

Topic subscription scalability fix (#1217)

Switching to using epoll system calls to improve performance of the topic polling code by a factor of 2. This required disabling the `addDelMultiThreading` test.
```
Polling Frequency

- **ROS Threads**

<table>
<thead>
<tr>
<th>Thread Name</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Main thread</td>
</tr>
<tr>
<td>PollManager</td>
<td>Publish/Subscribe I/O Polling</td>
</tr>
<tr>
<td>XMLRPCManager</td>
<td>Talk to rosmaster for connectivity map and services</td>
</tr>
<tr>
<td>ROSOutAppender</td>
<td>Logging</td>
</tr>
<tr>
<td>internalCallbackQueue</td>
<td>Connection drop, data store etc…</td>
</tr>
</tbody>
</table>

Polling Frequency Tunable for each processes, less CPU stress. (default freq is 100msec)

No Logging thread, less CPU stress. (instead of that, using own log system)
For ROS2

ROS2 Application can be switched the RMW implementation with environmental variable.

Under Consideration

Own RMW

Own Implementation
Special Thanks to ROS Community
SONYはソニー株式会社の登録商標または商標です。
各ソニー製品の商品名・サービス名はソニー株式会社またはグループ各社の登録商標または商標です。その他の製品および会社名は、各社の商号、登録商標または商標です。