A Lightweight and Portable Communication Framework for Multimodal Image-Guided Therapy

A. Schoch, B. Fuerst, F. Achilles, S. Demirci and N. Navab
Computer Aided Medical Procedures (CAMP), TU Munich, Germany

Motivation

- Multimodal and Intra-Operative Imaging (EndoTOFPET-US) [1]
- Collaboration of researchers from areas:
  - computer science (GUI, tracking, reconstruction)
  - physics (crystals, optics, attenuation and scatter correction)
  - medicine (tracers, pre-op imaging)
  - electronics (photo electronics, read-out, DAQ cards)
- Full monitoring and centralized control of system required

Goal

- Simple to use communication framework ("one header only")
- Plug-ins for visualization (CAMPvis)
- Open-source: https://campar.in.tum.de/Main/CAMPCom
- Control & Data Flow: communicate data and system messages between various components
- Efficiency: lightweight & efficient communication
- Platform Independence: serialization and transmission
- Abstraction & Extensibility: support extension, general interface
- Prioritization: asynchronous function to support efficiency

Inter-Device Communication

- Sample setup for EndoTOFPET-US Project
- Server handles connections based on subscription model

Query Handling & Prioritization

- Activity diagram for client library
- Simplified (without dispatcher thread pool)

Evaluation

Client/Server Communication Stability and Reliability

- 24h random data test
- No data loss, connection issues, memory leaks, or CPU loads
- Server memory approx. 1.4MB, CPU < 3%

Prioritization and Compression

- zlib based compression of MINI BITE dataset [3, 4]
- 67.60% compression \(\rightarrow\) overall transfer rate 3.35MB/sec
- 19.52% overall improvement compared to no compression

Transfer rates

<table>
<thead>
<tr>
<th></th>
<th>Loading data</th>
<th>Serialization</th>
<th>Sending</th>
<th>Receiving</th>
<th>Decentralization</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMPCom (Compression and Prioritization off)</td>
<td>16.464</td>
<td>6.074</td>
<td>11.258</td>
<td>0.612</td>
<td>27.311</td>
<td>0.612</td>
</tr>
<tr>
<td>Transfer rate [MB/sec]</td>
<td>10.785</td>
<td>8.883</td>
<td>8.052</td>
<td>0.728</td>
<td>24.385</td>
<td>0.728</td>
</tr>
<tr>
<td>OpenSUSE Link</td>
<td>150.039</td>
<td>8.129</td>
<td>113.399</td>
<td>113.399</td>
<td>403.152</td>
<td>1.2</td>
</tr>
<tr>
<td>Duration [sec]</td>
<td>16.464</td>
<td>13.269</td>
<td>11.258</td>
<td>0.612</td>
<td>27.311</td>
<td>0.612</td>
</tr>
<tr>
<td>Transfer rate [MB/sec]</td>
<td>10.785</td>
<td>8.883</td>
<td>8.052</td>
<td>0.728</td>
<td>24.385</td>
<td>0.728</td>
</tr>
</tbody>
</table>

Conclusion

- Stable, fast, reliable framework for IGT
- Easy integration, complex communication process hidden
- Similar performance compared to state of art
- Added features: prioritization, compression, subscription, thread pool, memory efficiency, simple integration