KIARA Demonstration
Context

FI-PPP
European programme for Internet-enabled innovation

FI-WARE
Technology Foundation of FI-PPP

KIARA
Advanced Middleware
KIARA - An advanced Middleware

- Message oriented communication middleware
- Simplifies communication between distributed heterogeneous systems
  - Systems with different operating systems
  - Applications using different programming languages and paradigms
- Abstracts network-layer with common API
KIARA - Advanced Features

- Data-structures used in an application can be dynamically mapped to IDL definitions at run-time
- No extra code generation (skeleton/stub) required to use middleware
- Embedded compiler (LLVM) generates highly optimized code
- Negotiation of optimal communication mechanisms, protocols, and data representations to be used between two peers
KIARA - Advanced Features

- Offers multiple communication paradigms like Request/Reply or Publish/Subscribe.
- Secure by Design approach: Applications can declare their security needs in the form of security policies (security rules) and apply them to data structures and service at development time or even later during deployment definitions.
### KIARA - First Benchmarks

<table>
<thead>
<tr>
<th>Localhost</th>
<th>Kiara ortecdr</th>
<th>BoostTyped</th>
<th>Thrift 0.9.1</th>
<th>Ice 3.5.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AvgLat</td>
<td>SSD</td>
<td>AvgLat</td>
<td>SSD</td>
</tr>
<tr>
<td>hurricane/L</td>
<td>10.74</td>
<td>0.08</td>
<td>21.40</td>
<td>1.31</td>
</tr>
<tr>
<td>hurricane/W</td>
<td>11.55</td>
<td>0.69</td>
<td>86.35</td>
<td>3.41</td>
</tr>
<tr>
<td>uragan/W</td>
<td>10.10</td>
<td>0.45</td>
<td>77.75</td>
<td>8.24</td>
</tr>
<tr>
<td>fpga/L</td>
<td>18.20</td>
<td>0.19</td>
<td>37.11</td>
<td>0.38</td>
</tr>
<tr>
<td>ghost/L</td>
<td>15.46</td>
<td>1.77</td>
<td>32.96</td>
<td>0.15</td>
</tr>
<tr>
<td>currywurst/L</td>
<td>21.43</td>
<td>0.18</td>
<td>39.40</td>
<td>0.21</td>
</tr>
<tr>
<td>bratwurst/L</td>
<td>13.29</td>
<td>0.16</td>
<td>24.11</td>
<td>0.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client → server</th>
<th>AvgLat</th>
<th>SSD</th>
<th>AvgLat</th>
<th>SSD</th>
<th>AvgLat</th>
<th>SSD</th>
<th>AvgLat</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ghost/L → fpga/L</td>
<td>178.23</td>
<td>6.73</td>
<td>190.73</td>
<td>5.10</td>
<td>195.67</td>
<td>10.50</td>
<td>182.67</td>
<td>8.79</td>
</tr>
<tr>
<td>hurricane/W → uragan/W</td>
<td>139.55</td>
<td>6.53</td>
<td>266.80</td>
<td>9.85</td>
<td>116.90</td>
<td>1.65</td>
<td>173.95</td>
<td>9.08</td>
</tr>
<tr>
<td>uragan/W → hurricane/W</td>
<td>130.40</td>
<td>9.05</td>
<td>254.25</td>
<td>12.36</td>
<td>127.50</td>
<td>6.38</td>
<td>169.15</td>
<td>8.81</td>
</tr>
<tr>
<td>currywurst/L → bratwurst/L</td>
<td>120.86</td>
<td>1.33</td>
<td>139.78</td>
<td>2.12</td>
<td>107.17</td>
<td>1.73</td>
<td>153.88</td>
<td>0.55</td>
</tr>
<tr>
<td>bratwurst/L → currywurst/L</td>
<td>120.35</td>
<td>0.45</td>
<td>142.49</td>
<td>0.99</td>
<td>108.16</td>
<td>2.09</td>
<td>152.12</td>
<td>1.23</td>
</tr>
</tbody>
</table>

- **Message-Size:** 470 Byte
- **Messages per Test:** 10,000
- **Measured Tests:** 20
- **Unit of Time:** micro-seconds
- **AvgLat:** average latency
- **SSD:** sample standard deviation
KIARA - Architecture
KIARA - Architecture Detailed View
KIARA - Responsibilities of ICCLab

- Offer transport capabilities to upper layers
- Offer functionality to negotiate transport, QoS and security parameters
KIARA - Transport Stack

Offers SCALN - a System Call Abstraction Layer for Networking

- bind / unbind
- connect / disconnect
- send / receive
- register_callback / callback
- get_context / set_context
- get_session / set_session
- get_configuration / set_configuration

Offers C-Bindings of SCALN

Support of additional languages planned
Transport Stack - Underlying Techn.

ZeroMQ: High-performance asynchronous messaging library aimed at use in scalable distributed or concurrent applications.

Why:
- very lightweight library
- implements several communication patterns that can be leveraged
- has very well defined clean documentation and API (reusable messaging stack)
Transport Stack - Underlying Techn.

**InfiniBand**: Switched fabric computer network communications technology used in high-performance computing and enterprise data centers.

**Why**:  
- For high throughput / low latency use cases  
- Offer RDMA (Remote Direct Memory Access)
**KIARA - Negotiation**

1. Set Local Capabilities

2. Send Offer

3. Compute Answer

4. Reply Answer

5. Communication starts
KIARA - QoS

- If the network has a SDN controller with the KIARA SDN application, we can assure/offer:
  - Bandwidth - certain amount of bandwidth with rate-limiters
  - Path - policy based privileged network paths
  - RTT - using monitoring to ensure max Round Trip Time
  - TOD - configure the communication according to the type of the devices along the path

- Transport with timestamp information (deadline) by using the Real Time Publish Subscribe (RTPS) protocol
Use different application types

```cpp
KT_Configuration conf;
conf.set_application_type(KT_STREAM);

KT_WEBSERVER);
KT_PUBLISHSUBSCRIBE);
KT_REQUESTREPLY);
```

Use different communication technologies

```cpp
KT_Connection* conn = new KT_Zeromq();

KT_InfiniBand();
KT_Boost();
```
KIARA Transport - Client Code Example

1. Create and configure connection

KT_Configuration config;
config.set_application_type (KT_REQUESTREPLY / KT_PUBLISHSUBSCRIBE);

config.set_host( KT_TCP, "192.168.100.1", 5555 );

KT_Connection *connection = new KT_Zeromq() / new KT_InfiniBand();
connection->set_configuration(config);

2. Use connection to send and receive messages

connection->connect(...);
connection->send(...);
connection->recv(...);
connection->disconnect(...);
3.1 Publish Request & Reply
Send PUT Request to http://160.85.4.249:8080
$ curl http://160.85.4.249:8080 -X PUT -d "some text"
We're Done.

Questions?

I got 99 problems

But my network ain't one.
Backup Slides
KIARA Negotiation Sequence

1: Server sets local capabilities

- Server sets local capabilities
KIARA Negotiation Sequence

2:  - Client sets local capabilities
    - Client composes offer
KIARA Negotiation Sequence

3: Client and Server negotiate transport and QoS settings
Internal Transport architecture

KIARA::Transport::KT_Connection
- _context : void*
- _socket : void*
- _session : std::map< std::string, KIARA::Transport::KT_Session >
- _configuration : KIARA::Transport::KT_Configuration*

+ KT_Connection()
  + KT_Connection( : string)
  + ~ KT_Connection()
  + connect(endpoint : KT_Client) : int
  + send(message : KIARA::Transport::KT_Msg& : linger : int = 0)
  + recv(linger : int = 0) : KT_Msg
  + disconnect() : void*
  + set_configuration(configuration : KIARA::Transport::KT_Configuration*)
  + get_configuration() : KIARA::Transport::KT_Configuration*
  + register_callback(callback : void (*)(KT_Msg*, KT_Session))
  + bind()
  + unbind()

KIARA::Transport::KT_Session
- _k_user_data : void*
- _endpoint : KT_Client
  + set_k_user_data(k_user_data : void*)
  + get_k_user_data() : void*
  + set_endpoint(endpoint : KT_Client)
  + get_endpoint() : KT_Client

KIARA::Transport::KT_Msg
- _metadata : std::map< std::string, std::vector< unsigned char > >
+ KT_Msg()
  + KT_Msg( : std::vector< unsigned char >& )
  + ~ KT_Msg()
  + add_metadata(key : string, : string)
  + get_metadata()
  + get_serialized_metadata( : string)
  + set_payload( : const std::vector< unsigned char > )
  + get_payload()

KIARA::Transport::KT_Configuration
- _network_layer : unsigned int = KT_IPELEGACY
  - _transport_layer : unsigned int = KT_TCP
  - _crypto_layer : unsigned int = KT_NONE
  - _application_layer : unsigned int = KT_ZEROMQ
  + _endpoint : KT_Client
  + _network_layer(network_layer : unsigned int = KT_IPELEGACY)
  + get_network_layer() : unsigned int
  + set_transport_layer(transport_layer : unsigned int = KT_TCP)
  + get_transport_layer() : unsigned int
  + set_crypto_layer(crypto_layer : unsigned int = KT_NONE)
  + get_crypto_layer() : unsigned int
  + set_application_layer(application_layer : unsigned int = KT_ZEROMQ)
  + get_application_layer() : unsigned int
  + set_application_type(application_type : unsigned int)
  + get_application_type() : unsigned int

KIARA::Transport::KT_Client
- _endpoint_identifier : string
  + set_endpoint(endpoint : string)
  + get_endpoint() : string
Because nobody wants to write all this for just a webserver

```c
kt_configuration_t* config = kt_configuration_create();
kt_configuration_set_network_layer( config, KT_IPLEGACY );
kt_configuration_set_application_layer( config, KT_ZEROMQ );
kt_configuration_set_transport_layer( config, KT_TCP );
kt_configuration_set_application_type( config, KT_REQUESTREPLY );
kt_configuration_set_hostname( config, "localhost" );
kt_configuration_set_port( config, 5555 );
```
KIARA - Negotiation

Goal: Negotiate application and communication settings

1. Both, Initiator (client) and target (server) set their local capabilities

2. Initiator composes an offer out of his local capabilities and sends an offer to the target

3. The target compares the remote capabilities with the local ones and sends back an answer

4. The initiator sets the negotiated answer as final and starts the communication

Optional step 2: The initiator fetches the capabilities of the target