### Background

- Power grid operational data are exchanged in the power grid between different entities.
- Gateways are used to facilitate the exchange of different types of operational data.
  - Examples: ICCP Gateways, Phasor Gateways
- Gateways need to process and/or translate operational data between different protocols.
  - These protocols need to be decomposed to a standard intermediary representation, as different protocols bundle data differently.
  - Decomposes to \{Timestamp, GUID, Value\} (aka a Measurement).
- As the grid moves towards a distributed and in-network processing paradigm, an efficient and secure communication mechanism is needed between member nodes or gateway modules spread across a cluster of physical machines.

### Goals

- To build a flexible & secure messaging middleware for gateways (particularly phasor gateways) with:
  - Ability to leverage multiple COTS messaging systems.
  - Built-in access control to provide Measurement-level granularity.
- Design a standard API to encompass the different messaging systems.
- Build access control on top of messaging systems.

### Fundamental Questions/Challenges

- Design a standard API to encompass the different messaging systems.
- Build access control on top of messaging systems.

### Research Plan

- Investigate COTS messaging systems, paying particular attention to:
  - Brokers, Communication, Authentication, Access Control, Speed, and API “niceness”
- Design a middleware layer that can support multiple COTS messaging systems.
- Implement and document the API for our middleware layer.
- Design an access control layer for our middleware.

### Message System Evaluation

<table>
<thead>
<tr>
<th>Messaging System</th>
<th>Broker Communication</th>
<th>Supported</th>
<th>Authentication</th>
<th>Access Control</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSMQ</td>
<td>Distributed and Centralized</td>
<td>Inter-process</td>
<td>Per queue</td>
<td>Store-and-forward Per queue</td>
<td>Faster than SQL Message broker</td>
</tr>
<tr>
<td>JMS</td>
<td>Distributed and Centralized</td>
<td>Inter-process</td>
<td>Per connection Per queue Per topic</td>
<td>System-level (DS-level) Destination-level (queue and field level)</td>
<td></td>
</tr>
<tr>
<td>OWQ</td>
<td>In-process, Inter-process, TCP, multicast</td>
<td>Topic based</td>
<td>IP Address and/or user name &amp; password</td>
<td>System-level (DS-level) Destination-level (queue and field level)</td>
<td></td>
</tr>
<tr>
<td>DDS</td>
<td>Distributed</td>
<td>Topic based</td>
<td>Hostname, user name &amp; password, Port number</td>
<td>System-level (DS-level) Destination-level (queue and field level)</td>
<td></td>
</tr>
<tr>
<td>AMQP: Rabbit MQ</td>
<td>Distributed</td>
<td>TCP, Inter-host</td>
<td>Hostname, user name &amp; password, Port number</td>
<td>System-level (DS-level) Destination-level (queue and field level)</td>
<td></td>
</tr>
</tbody>
</table>

### Middleware Design & Implem.

- Two messaging systems have been targeted for initial implementation:
  - AMQP: Rabbit MQ
  - MSMQ
- System architecture:
  - Gateway System
  - COTS Message Queues
  - Message Bus Layer
  - Power Grid Protocols

### Future Work

- Tune the API to allow simple integration of new middleware systems in the future.
- Access control layer for all the APIs.