Processing Real-time Sensor Data Streams for 3D Web Visualization

Arne Bröring*, David Vial*, Thorsten Reitz**

* Esri Suisse
** Esri R&D Center Zurich

5th IWGS Workshop @ ACM Sigspatial
4th November 2014, Dallas, USA
We are more & more surrounded by sensors...
**Aim:** Gaining knowledge from big sensor data

→ Web-based

3D Visual Analytics for real-time sensor data

• **Challenge:**
  > High data rates and volumes.
  > 3D engines (e.g. WebGL) render frames using *strict time budget*.

→ 3D engines have performance & scalability issues, when: directly pushing high frequency & irregularly data
**Aim:** Gaining knowledge from big sensor data

**Goal:**

- process sensor data streams to enable efficient data transmission to 3D visualization clients.
Approach

- **Design of:**
  1. *event-driven architecture* to process sensor data streams
  2. *processing patterns* for efficient transmission of sensor stream data to 3D clients
Architecture

3D Visualization Client

- Pulls 3D scene

Geospatial Data Server

- Registers at
- Forwards sensor data to

Sensor Data Push Service

- Publishes messages to

Sensor Data Broker

- Pushes features to

Feature Push Service

3D Scene Download Service

- Pulls 3D scene

Sensor

Sensor Data Push Service
Architecture Implementation

3D Client (e.g. Web Scene Viewer)

- Pulls *i3s* scene

Scene Service

- Registers at
- Forwards sensor data to *topic*

ArcGIS Server

- Publishes *i3s* features to

GeoEvent Processor

Sensor

IBM Intelligent Operations Center

MQTT Broker

© 2014 Esri Suisse
i3s Format

- Stands for: Indexed 3D Scene
- Newly developed by Esri
- Format to stream 3D GIS data to mobile, web and desktop clients
- JSON encoded
- Default format of the ArcGIS Scene Service
- Clients:
  - ArcGIS Web Scene Viewer
  - ArcGIS Mobile Runtime
  - ArcGIS Pro
MQTT

- Stands for: *Message Queue Telemetry Transport*

- Messaging protocol to realize **pub/sub** on top of the TCP/IP

- **Light-weight**; for limited processing power and/or network bandwidth

- Pub/sub requires **message broker**

- Broker distributes messages to clients based on **topics**
Patterns for Sensor Data Stream Processing
Sensors & Features

- High frequency in data
- Irregular data sending
- Large / small features (variety in characteristics)
• Irregular data transmission from server to client
• Inefficient, e.g., when large features are mostly constant, but one characteristic changes
Pattern 1: Aggregation over Time Interval

- Periodical data transmission from server to client
Pattern 2: Sending Latest Feature Only

- Efficient, e.g., when very high numbers of messages/features are coming in to the server

→ Good if client only needs latest representation
Efficient when features have a lot of changes on single characteristics, while others remain constant.

→ Good if client only needs the differences
Evaluation

• Configuration in GeoEvent Processor administrator:

- Message Separator: \n
- Attribute Separator: ;

- URL Path: /I3S-WebSocket

- Layer Name: layer

- Update Interval (seconds): 1

- Send Latest Only: false

- Send Diff Only: true

- Host: localhost

- Port: 1883

- Topic: SBC/test
Evaluation

- Test client for:
  - Sending **MQTT** messages
  - Receiving **i3s** messages
Evaluation

- Automatically generated 4 CSV files for:
  - 10 features, each having 10 attributes
  - 100 update cycles
  - 20%, 40%, 60%, & 80% likelihood of change in feature characteristics
Evaluation

• Direct Data Transmission
  > Original CSV file size: 275.108 bytes
  > Received i3s file size: 1,282,878 bytes
  > # data pushes: 1,000 (10 features x 100 updates)

• Pattern 1 - ‘Aggregation over Time’

<table>
<thead>
<tr>
<th>Update Interval</th>
<th># Pushes</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 s</td>
<td>48</td>
<td>1,411,955 bytes</td>
</tr>
<tr>
<td>2 s</td>
<td>24</td>
<td>1,408,901 bytes</td>
</tr>
</tbody>
</table>
Evaluation

- Pattern 2 - ‘Sending Latest Feature Only’

<table>
<thead>
<tr>
<th>Update Interval</th>
<th># Pushes</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 s</td>
<td>11</td>
<td>157.234 bytes</td>
</tr>
<tr>
<td>2 s</td>
<td>6</td>
<td>85.926 bytes</td>
</tr>
<tr>
<td>3 s</td>
<td>4</td>
<td>57.289 bytes</td>
</tr>
<tr>
<td>4 s</td>
<td>4</td>
<td>43.001 bytes</td>
</tr>
<tr>
<td>5 s</td>
<td>3</td>
<td>42.985 bytes</td>
</tr>
</tbody>
</table>

- 1 % of # pushes
- 12% of the data size

Within defined time period, server stores latest version of feature, which is sent to client.
Evaluation

- Pattern 3 - ‘Sending Feature Differences Only’

<table>
<thead>
<tr>
<th>Change Likelihood</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 %</td>
<td>310.661 bytes</td>
</tr>
<tr>
<td>40 %</td>
<td>541.135 bytes</td>
</tr>
<tr>
<td>60 %</td>
<td>773.601 bytes</td>
</tr>
<tr>
<td>80 %</td>
<td>975.736 bytes</td>
</tr>
</tbody>
</table>

24% of the data size

- Server maintains a most recent representation of feature and sends changed feature characteristics
- Higher change likelihoods → bigger data size
Conclusions

→ Reference architecture + 3 processing patterns

> Pattern (1) ‘Aggregation over Time’
  > High reduction in: # data pushes
  > Good if client needs all information in periodical updates

> Pattern (2) ‘Sending only Latest Feature Representation’
  > **Highest** reduction in: # data pushes & data size
  > Best if client does not need all information

> Pattern (3) ‘Sending only Differences in Characteristics’
  > High reduction in: data size
  > Good if client needs all information