Weather Monitor System (WMS) –

Communication Protocols

# 1 Overview

WMS includes four different kinds of processes: Weather Stations (WS), Data Servers (DS), Forecast Servers (FS), and Weather Client (WC). This document defines how these processes communicate with each other. Table 1 provides a summary list all the protocol that govern and the remaining sections define these in more detail. These communication protocols assume that all host machines, including those that will run Weather Stations and Weather Client, will join a common virtual private network.

**Table 1 – Protocol List**

| Purpose | Initiator | Other Processes | Pattern |
| --- | --- | --- | --- |
| Discovery of Data Servers | WS, FS, WC, DS | DS | Multicast with Point-to-Point reply |
| Discovery of Forecast Servers | WC | FS | Multicast with Point-to-Point reply |
| Publish Weather Data | WS | DS | Point-to-Point (Unreliable) |
| Get Weather Data | FS, WC | DS | Request Reply with 2nd Data Channel |
| Get Forecast | WC | FS | Request Reply with Immediate State Pattern |
| Weather Data Sync | DS | DS | Request Reply with 2nd Data Channel |

# 2 Messages and Shared Objects

See Figure 3 and 4 in the Architecture Design for the details on the kinds of messages that the system’s applications communication protocols will use.

# 3 Communication Patterns

The protocol for WMS will make use of the following Application Communication Protocol (ACP) idioms and patterns, as defined by CommDP at http://commdp.serv.usu.edu.

Point-to-Point Send (P2P)

Multicast (MC)

Request Reply (RR)

Intermediate State Message (ISM)

Second Channel (2C)

# 4 Communication Protocols

**4.1 Discovery of Data Servers**

Any process in the system can discover available Data Servers using a hardware multicast and waiting for replies within a given time frame.

Message Sequence and process behaviors:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seq #** | **Sending Process(es)** | **Receiving Process** | **Transport Protocol / Communication Channel** | **Message** | **Process Behavior** |
| 1 | A = any process | B = any DS | Group Multicast to 224.1.1.2, Port 5000 | Server Discovery | When B receives the message, it will immediately response back with a Server Alive message |
| 2 | B | A | UDP, using endpoint from which the first message was sent | Server Alive | Process A believes that B is operational and may start other conversations with B. |

**4.2 Discovery of Forecast Servers**

Any process in the system can discover available Forecast Servers using a hardware multicast and waiting for replies within a given time frame.

Message Sequence and process behaviors:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seq #** | **Sending Process(es)** | **Receiving Process** | **Transport Protocol / Communication Channel** | **Message** | **Process Behavior** |
| 1 | A = any process | B = any DS | Group Multicast to 224.1.1.3, Port 5001 | Server Discovery | When B receives the message, it will immediately response back with a Server Alive message |
| 2 | B | A | UDP, using endpoint from which the first message was sent | Server Alive | Process A believes that B is operational and may start other conversations with B. |

**4.3 Publish Weather Measurement**

Any WC can publish weather data to a known DS using this protocol, which leverages both the RR and 2C ACP patterns. A RR exchange of message take place on a UDP-based channel to start the conversation and exchange information about how the setup the second channel. The DS will transmit the weather data on the second channel.

Message Sequence and process behaviors:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seq #** | **Sending Process(es)** | **Receiving Process** | **Transport Protocol / Communication Channel** | **Message** | **Process Behavior** |
| 1 | A = any WC | B = any DS | UDP, port 5002 | Weather Data Message | When B receives the message, it stores it in a persistent place, like a database. |

**4.4 Get Weather Data**

Any FS or WC can request a slice of weather from any DS using this protocol, which leverages both the RR and 2C ACP patterns. A RR exchange of message will take place on a UDP-based channel to start the conversation and exchange information about how the setup the second channel. The DS will then transmit the weather data on the second channel.

Message Sequence and process behaviors:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seq #** | **Sending Process(es)** | **Receiving Process** | **Transport Protocol / Communication Channel** | **Message** | **Process Behavior** |
| 1 | A = any FS or WC | B = any DS | UDP, port 5002 | Weather Data Request | When B receives the message, Process B creates a TCP communication channel and begins listen for a connection on that channel. Process A wait for a Data Channel Info message in response to the request. |
| 2 | B | A | UDP, using endpoint from when sent the first message | Data Channel Info | Process A connect to the B using a TCP socket. On connection, B processes original request. Specifically, if the request is not a recent duplicate message, B should compute the requested data slice. If the request is a recent duplicate, B lookup the results previous request. |
| 3… | B | A | TCP, on 2nd channel | Weather Data Message | Process B sends as many Weather Data messages as needed to communicate the entire data slice.  After sending all the necessary Weather Data Messages, B closes the 2nd channel and ends the conversation. Process A should catch the closure event and treat that as the end of the conversation. |

**4.5 Get Forecast**

Any WC can request a 10-day forecast from Forecast Server from any FS using this protocol, which uses the RR pattern with an IMS pattern.

Message Sequence and process behaviors:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seq #** | **Sending Process(es)** | **Receiving Process** | **Transport Protocol / Communication Channel** | **Message** | **Process Behavior** |
| 1 | A = WC | B = any FS | UDP, port 5002 | Forecast Request | When B receives the message, Process B send back a Progress Status with an initial status of 0% complete, begins to compute the requested forecast |
| 2…x | B | A | UDP, using endpoint from when sent the first message | Progress Status | Every 2 seconds, B sends a Process Status message back to A with an estimated percent complete |
| x+2 | B | A | UDP, using endpoint from when sent the first message | Weather Data Message | Process B sends a Weather Data Message containing exactly 10 Weather Data object, one for each day at the specified hour of the day. |

**4.6 Weather Data Sync**

Any DS may request replicas of weather data from any other DS using this protocol, which leverages both the RR and 2C ACP patterns. A RR exchange of message will take place on a UDP-based channel to start the conversation and exchange information about how the setup the second channel. The DS will then transmit the weather data on the second channel.

Message Sequence and process behaviors:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Seq #** | **Sending Process(es)** | **Receiving Process** | **Transport Protocol / Communication Channel** | **Message** | **Process Behavior** |
| 1 | A = any FS or WC | B = any DS | UDP, port 5002 | Sync Request | When B receives the message, Process B creates a TCP communication channel and begins listen for a connection on that channel. Process A wait for a Data Channel Info message in response to the request. |
| 2 | B | A | UDP, using endpoint from when sent the first message | Data Channel Info | Process A connect to the B using a TCP socket. On connection, B processes original request. Specifically, if the request is not a recent duplicate message, B should compute the requested data slice. If the request is a recent duplicate, B lookup the results previous request. |
| 3… | B | A | TCP, on 2nd channel | Weather Data Message | Process B sends as many Weather Data messages as needed to communicate the entire data slice.  After sending all the necessary Weather Data Messages, B closes the 2nd channel and ends the conversation. Process A should catch the closure event and treat that as the end of the conversation. |