

HUB PP System Block Diagram

**General System Description**

The HUB PP is a server-based application, which handles all the HUB’s logical and data processing functionality. It connects tothe HUB’s HW modems modules, over ethernet. The HUB PP is talking to various servers on the Internet that includes the hiSky NMS/OSS cloud services and customer’s IoT (App.) data servers. On the satellite side, the HUB PP prepares data packets and converts them to TSs (Time Slots) to be transmitted by the HUB modems to the Terminals,and receives TSs that were transmitted by the Terminals thru the satellite and converts them to UDP packets.

**IP addressing and UDP Ports Scheme**

IP and UDP wise, the communication between the Sensor that is attached to the terminal and the App server on the Internet, is done as described here:

1. Assumption: only one sensor is attached to each terminal.
2. The Sensor “knows”(by configured of the sensor)the (static) IP address of the terminal and the UDP port of the service on the terminal that it should connect to.
3. The HUB PP knows for each terminal/service/sensor who is the App server that is serving this sensor. This is done by reading a configuration file (or from NMS) that links the(long) terminal number + port to an App server’s hostname/IP address and destination UDP port.
4. The HUB also knows the sensor/service UDP source port (which is sent as source port from the terminal in each packet from this sensor/service).To uniquely identify the sensor and the terminal by this number, when communicating with the App server, the HUB needs to replace this port number with another port number. See an example in the tablebelow:

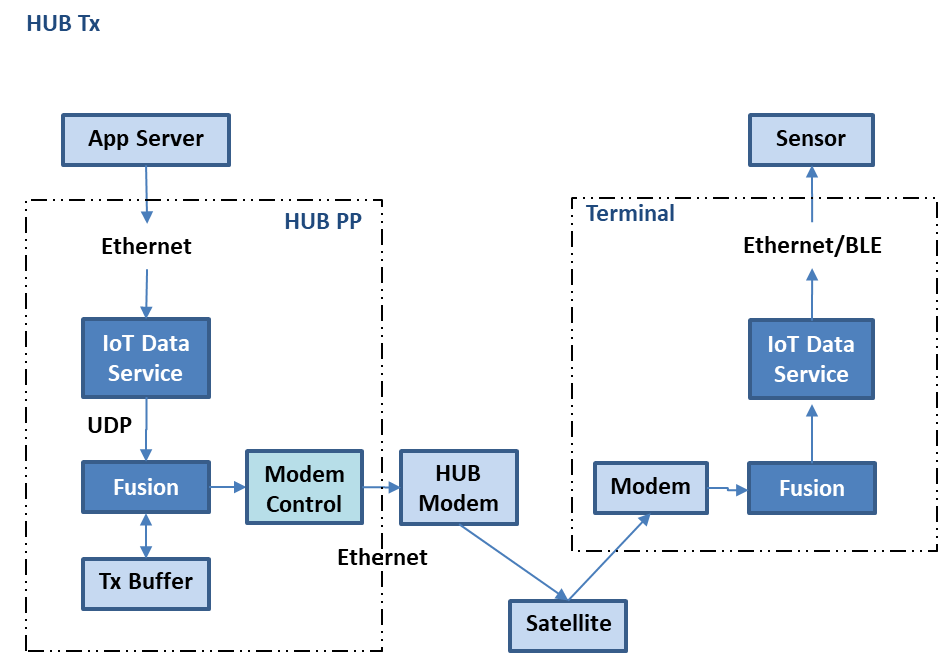
|  |  |  |
| --- | --- | --- |
| Terminal ID | Service Port | Unique Port for App Server |
| 12 | 2200 | 10200 |

1. When the HUB sends a packet to the app server, it will use the unique port (from the table above) number as the source port.
2. When the hub sends a packet to the sensor/service it will use the service port as the destination port.
3. The data flow is as follows:
   1. The sensor sends a UDP packet to the terminal’s IP address and the service’s UDP port. This packet also bears the source port of the sensor.
   2. The terminal takes the L3/L4 headers off the packet and sends it to the HUB PP. The UDP source port of the sensor is also sent to the HUB inside the stripped packet.
   3. The HUB PP gets the packet from the terminal, reassemble the L3/L4 headers using the IP address of the App server as destination, the UDP port of the App server as destination port and a theunique UDP port from the table above as the source port that identify the relevant service/sensorwithin the HUB PP domain.
   4. The App server will send a reply packet using the HUB PP IP address and the UDP destination port that it received from the HUB PP as source port.
   5. The HUB PP will use the destination port as the index for sending to therelevant service/terminal port according to the table above andwill send the packetafter taking off the L3/L4 headers.

**Data Path General Description**

HUB Tx Path

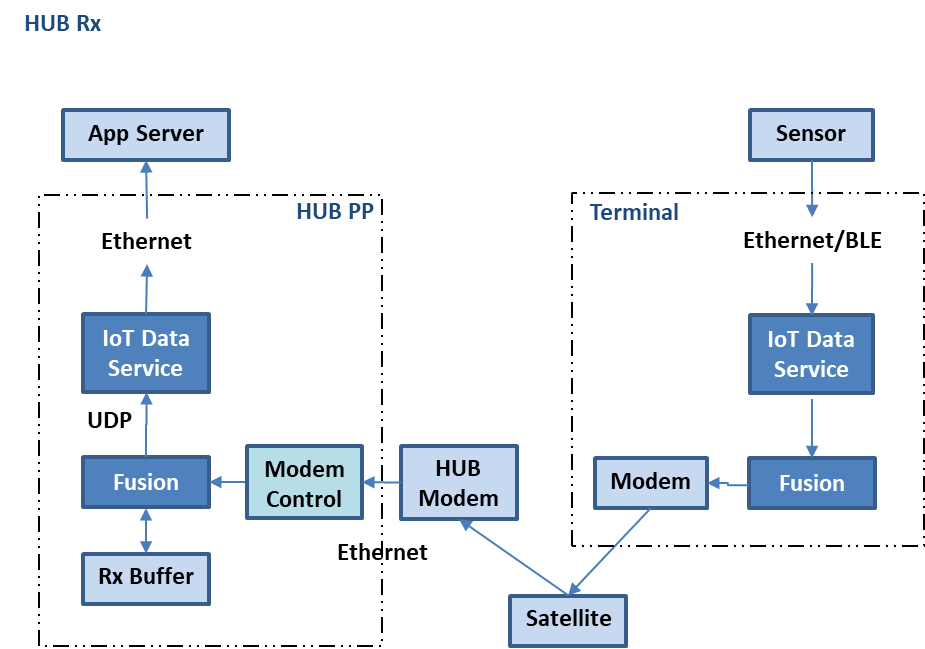
The following diagram describes the path of a data packet that is sent from an application server on the Internet to an IoT sensor that is connected to a hiSky Terminal.



A UDP packet that is sent from the App server, which resides on the Internet, is sent via a UDP socket connection to the IoT Data service module on the HUB PP. The destination on the UDP packet is the HUB PP address.The destination UDP port identifies the relevant service on the HUB PP. The packet is received by the IoT data service in the HUB PP. The service then stripes the packet form its original L3/L4 headers, puts new headers and sends the packet using anotherUDP socket connection with the Fusion module within the HUB PP. When the fusion gets the packet, it removes the L3/L4 headers and determine if a fragmentation is required, based on the length of the packet payload and the defined length of the satellite TS. If a fragmentation is required, the fusion will fragment the packet to a few TSs, add a “fusion header” (see fusion description) to each TS and send the TS to the Modem Control unit.

The modem control module sends the TS(s) to the HUB Modem unit over a UDP socket that runs over an Ethernet connection. This connection sends an ack message for every transmitted packet and also has a backpressure option, forsignalinga Tx congestion status. The Hub Modem sends the packet over the satellite to the terminal. In the terminal theRx path is used to restore the original UDP packet by the MACand send it to the sensor via the terminal’s fusion and the relevant service.

HUB Rx Path



The HUB Rx path is described in the figure above. A UDP packet that is sent from the IoT sensor on the terminal side, is received by the terminal on a socket connection over Ethernet or BLE. In the terminal, the packet is received by the IoT data service and transferred to the fusion andMAC. The MACstrips the L3/L4 headers from the packet and converts the UDP packet into hiSky air protocol TSs(the packet is also fragmented into a few TSs if required) and sends it over the satellite link. The Packet is then received by the HUB modem, which packs it inside an UDP packet and send it over a UDP socket to the HUB PP modem control unit. The modem control unit hands the received packet to theFusion and sends an ack packet to the HUB Modem unit.

The Fusion puts the received packet into the Rx buffer of the terminal that sent the packet. If the original packet was fragmented into a few fragments (TSs) by the MAC in the Terminal, each fragment that is received by the HUB, is then put into the terminal’sRx buffer in a FIFO fashion.Thereceived packet(s) in the Rx buffer of the relevant terminal, also has a tag that identifies the service that sent the packet. Based on this information, the HUB Fusion unit reads the buffer. If a fragmented packet was received, the Fusion will wait until all fragment are received. The Fusion then re-constructsthe original UDP packet and sends it to the relevant IoT service, over an internal UDP socket connection. The relevant service then sends the packet to the app server on the Internet using an external socket connection.

**The Core Admin**

The Core Admin is responsible for the management of the HUB PP high-level operations. The Core Adminmanage the following functions:

1. Initialize the HUB PP system.
2. Register and authenticate the HUB PP with the NMS system.
3. Register and authenticate each new terminal according to a local file, or the NMS info, and assign each registering terminal a short terminal ID number.
4. Collect operational and error data from the HUB modems, the HUB PP and the terminals and log them locally and send them to the NMS.
5. Send the PHY initiationparameters file to the PHY control module. The relevant PHY parameters will be based on information from a local file or the NMS.
6. Get the configuration information for each terminal from a local file or the NMS.
7. Create the required service(s) for each terminal based on its usage profile.
8. Generate the TS Assignment table and send it to the Fusion.
9. Maintain and handles the Active Terminal list.
10. Connect to the NMS server.
11. Handles the telemetry messages (need definition).

System startup and Registration of the HUB PP

When the HUB PP system starts, the first module that runs is the Core Admin. The first task of the Core Admin is to register the HUB PP itself with the NMS service. The details for registration will be taken from a local information file that will contain the following:

1. System serial number.
2. System SW version.
3. System HW details.
4. Linux OS details.
5. System location details.
6. Other TBD.

The detailed registration process with the NMS is TBD.

A registration request that contains all the details will be sent to the NMS. The NMS will send a registration reply message that approves or denies the HUB PP registration. If the registration is approved, the operation continues. Else, the operation will stop, and an appropriate message will be displayed on the Technician GUI.

Terminal Registration

When a terminal is initialized, it sends a registration request packet to the HUB PP. The structure of this packet is described in the hiSky air protocol description document. When a registration request is received in the HUB PP, it is transferred to the Fusion. The fusion identifies that this is a Registration Request according to the opcode (0x01) of the message. The fusion then transfers the message to the Core Admin. The Core Admin is following the below procedure:

1. If the number of active terminals is more then max\_terminals (400 for the MVP), then the registration request is denied and a registration reply message is sent to the terminal with the approve/reject code = 0 (see air protocol doc.).
2. Else, the Core Admin will search for the Terminal HUID in the list of active terminals. If the terminal is in the list, the Core Admin will issue a Registration Reply and indicate in the table that the terminal is active.
3. If the terminal is not listed in the active terminals list, the Core Admin will add the terminal to the list, mark it as active and send a Registration Reply message to the terminalwith the Approve/Reject code = 1.
4. The Core Admin adds the terminal to the active terminals list (see below).
5. The Registration Reply message alsocontains a short terminal ID number (12 bit). This number will be derived from the active terminals table as the next unused number on the list (last short ID number used + 1). If the last number is 4095, then the new ID will be 1.

Monitoring Terminal activity

When a terminal is first registered with the HUB PP, it will be added to the active terminals table that is shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| HUID | Short ID | Active Since | Last Activity |
| 6 Bytes | 12 bits | Timestamp | Timestamp |

Active Terminals Table Format

The Core Admin will maintain the table in the following manner:

1. The core admin will get periodic “*updatemessage*” from the fusion that summarizes the HUB PP traffic ((see *Fusion* description). The data from this message will be used to update the Active terminals table.
2. When the *Core Admin* gets an update message from the *fusion* it will update the “*last activity*” field of the table for each terminal that appears in the update message.
3. Every defined (by a configuration file) active\_terminals\_scan\_cycle, the Core Admin will:
   1. Go thruall the table.
   2. If for a certain terminal,the time elapsed from the last activity is more than a defined (by a configuration file) active\_terminal\_tiemout, then the terminal will be deleted from the table.
   3. An inactive terminal will be removed from the assignment table.
4. Comment for terminal behavior and ICD: If a terminal does not appear in the TS assignment table, and it is still active, it should send a new registration request.

TS Assignment Table

The Core Admin will generate a TS (Time Slots) Assignment table that will be transmitted to the terminal in a certain time cycle. The table will include the number of TSs per terminal and the number of TSs for the Outbreak option, in which a new terminal can send a registration request or other urgent message.

The behavior of the terminal when acting upon the table is described in a separate file.

The following definition are used in the description of this functionality:

1. ts – time slot.
2. ts\_per\_sec – Number of TSs per second. This is a defined constant.
3. Interval – Cycle time in seconds. Within the time of each Interval, all the active terminals are getting TS assignments (minimum one TS for data Tx per terminal).
4. ts\_size – Bytes per TS.
5. active\_terminals\_# – The number of current active terminals.
6. outbreak\_ts\_# – Number of TSs that are allocated for the Outbreak period. For the MVP this number will be 40.
7. table\_tx\_cycle – the time period between transmissions of the table. 2 seconds for the MVP.
8. table\_assignmet\_period – the time period of assignments in the table. 10 seconds in the MVP.

The table will be transmitted every table\_tx\_cycle (2 seconds in the MVP) and will always include the TS assignment for the next table\_assignmet\_period (10 seconds in the MVP).

Comment: The table assignments are actually given in a sliding window. The assignments period overlap between consecutive tables transmissions is table\_assignmet\_period minus table\_tx\_cycle. For the MVP this is 2 seconds. This means that for the MVP, a consecutive table will repeat 8 seconds of the assignments of the former table and only 2 seconds will be new assignments.

The actual table format to be sent over the air, is described in the air protocol doc. Depending on the number of active terminals,The table may need to be spread over more than one TS.

The following table shows the interval in secondsfor each number of active terminals:

|  |  |
| --- | --- |
| active\_terminals\_# | Interval |
| 1-99 | 20 |
| 100-199 | 45 |
| 200-299 | 60 |
| 300-400 | 100 |

For the MVP, the assignment of TSs per terminal will be calculated as follows:

FLOOR (((ts\_per\_sec \*Interval) -outbreak\_ts\_#)/active\_terminals\_#)

Example

For the MVP, with 30 active terminals the tables will look like this:

FLOOR (((20\*20) -40)/30) = 12

Table 1

|  |  |  |
| --- | --- | --- |
| Terminal ID | Assignment |  |
| 1 | 12 |  |
| 2 | 12 |  |
| 3 | 12 |  |
| … | … |  |
| 9 | 12 |  |
| 10 | 12 |  |

Services creation and deletion by the core admin

The core admin creates (and deletes) theHUBPPservices. When the core admin gets a registration request from a terminal (thru the fusion), the core admin uses the (long) terminal ID to get the services details for the requesting terminal from a local configuration file (or the NMS). The format of the file is as follows:

|  |  |  |
| --- | --- | --- |
| Terminal long ID (HUID) – 6 Bytes | Service type- 1 Byte |  |
| 1234 | 1 |  |
| 1234 | 2 |  |
| 1235 | 1 |  |

The Core admin creates the required services and assigns each service with a unique service ID and two port numbers – one for the internal connection to the fusion, and one for the connection to an external App server. The ID and port numbers are maintained as a *service table* of the Core Admin, with the following format:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Terminal Long ID | Terminal Short ID | Service ID for the fusion | Service source port for Fusion connection | Service source port for App server connection |  |
| 6 Bytes | 12 bits | 2 Bytes | 1025 - 65535 | 1025 - 65535 |  |
| … |  |  |  |  |  |

The initial values for the above table will be taken from a configuration file with the following format:

Initial short terminal ID: 1

Initial service source port for Fusion connection: xxxx

Initial service source port for App server connection: xxxx + 1000

If the Core Admin decides that a terminal has turned inactive (see below), the Core Admin will delete all the services that are related to this terminal in the HUB PP system.

**The Service(s)**

General

For the MVP version, two (UDP and ERM sensor) types of user data services will be supported. These data services are transferring point to point UDP packets between the Fusion and the external App servers on the Internet. The services will use a UDP socket-based connection to talk with the Fusion unit. The service will also use another socket-based connection to talk to the relevant customer external App. server on the Internet.

All the details of the UDP ports and addresses that are required for the creation and operation of the service will be given upon creation of the service by the Core Admin module. The Core Admin takes these parameters from a local configuration file or the NMS.

Service Creation and Deletion

For each sensor a service is created. The service is created by the Core Admin. Each new service is created with a unique service ID number. A Service is active as long as the terminal it is working with is active. If the terminal is considered inactive, the service will be deleted (exit/return). When the service is created, it will first establish a socket connection as a client, with the app server on the Internet, using the hostname/IP address and the UDP port for this server, and a source port. The address/Hostname and port numbers are given to the service by the Core Admin upon creation.

Service Data Flow

Each service is connected to two sockets – One with an Application server on the internet and one with the fusion. Both connections are UDP based.

**Service Tx flow**

Each time a service gets a UDP packet from the App. Server on the Internet, it will remove the L3/L4 headers, and use the payload to build a new UDP packet that will be sent over the socket with the fusion, for transmission to the terminal.

**Service Rx flow**

Each time a service gets a UDP packet from the Fusion, it will remove the L3/L4 headers, and use the payload to build a new UDP packet that will be sent over the socket that the service has with the relevant app server on the Internet.

**The Fusion**

The fusion module concentrates all the satellite data traffic to and from the HUB PP. It has the following functions:

1. Monitor Tx and Rx traffic and collect statistical data.
2. Prepare and send an “Update message” for the Core Admin.
3. Allocate and delete buffers for Tx and Rx as required.
4. In Tx, if the packet is too long to fit in one TS, the fusion will fragment the packet into a stream of TSs and will send the TSs to the Modem Control interface unit.
5. Get packets for transmission from the user services or the core admin, remove L3/L4 header, encapsulate the data in TS format and send the TSs to the Modems Control unit.
6. Get Rx TSs from the Modem Control unit. If the TSs are fragmented, defragment the TSs to build the original message.
7. Send the Rx messages to the relevant Service or the Core Admin.
8. Handle the modems control protocol.
9. Apply back pressure control over the interface with the modem control as required.
10. Assignment table insertion in constant TS periods.

**Transmission of TSs, Fragmentation and Tx Buffers handling**

The Fusion will get packet for Tx from the services and from the Core Admin. For each packet the following logic will apply:

1. The packet is striped from the L3/L4 headers.
2. If the packet’s payload size is smallerthanthe TS size, then the TS header is added to the payload and the TS is sent to the Modems.Any empty space in the TS payload will be padded by zeros.
3. If the packet’s payload size is larger than the TS size, the packet payload is put in a Tx buffer of the relevant terminal. The fusion will keep a pointer to the last byte of the payload that was already sent.
4. Each time the Fusion needs to send a TS to the relevant Terminal,it will take part of the payload to fill the TS payload according to the “last sent” pointer and update the pointer as required.

The Fusion will create a Tx buffer for each service and for the Core Admin and the assignment table. The Fusion will decide which Tx buffer will get a priority when selecting packet for Tx. See below.

If a terminal is declared inactive by the Core Admin, the Fusion will delete the buffers of this terminal.

**Selection of TSs for Tx**

The Fusion will select the TSs for transmission according to the following algorithm:

1. The Fusion will send TSs to the modems in a “round robin” fashion, as long asthe modems do not issue a back-pressure message.
2. The selection of TS for Tx will be based on TS count.
3. Every air\_interval TS count, a new assignment table will be taken from the assignment table Tx buffer and sent in one or more TSs.
4. After the assignment table Tx, the fusion will take TSs from the available Tx buffers.
5. If a packet is fragmented, the Fusion will send all the fragments and then will start sending another packet to the next terminal.
6. If a TS is not filled by one packet, the Fusion may put more then one packet in a single TS.
7. If the buffers are empty, the fusion will send a zero padded TS.
8. If a backpressure signal is received from the modems unit, the fusion will stop sending TSs until the backpressure is released.

**Receiving of TSs,Reassembly and Rx buffers handling**

The Fusion will receive the TSs from the Modem Control unit in the following procedure:

1. A TS is received by the fusion.
2. If the TS is a fragment, then the fusion will put the TS in the Rx buffer of the relevant service.
3. If the TS is the last fragment and all fragment have been received, then the fusion will reconstruct the full packet and send it to the service or the core admin.

For each service and for the Core Admin, the fusion will create an Rx buffer. When a service is deleted due to terminal’s inactivity, the fusion will delete its Rx buffer.

**Tx and Rx Statistics Collection**

The Fusion will collect statistic regarding Rx and Tx data flow. Zero padding should not be counted. The information will be collected per each terminal and put in a table per required time periods.

The following table will be used to store the statistical data:

|  |  |  |  |
| --- | --- | --- | --- |
| Terminal long ID | Time stamp | Rx | Tx |
| 6 Bytes | DD/MM/YYYY hh:mm:ss (14 Bytes) | 4 Bytes | 4 Bytes |

Every time period or number of lines (TBD) this table will be reset and saved in a local log file (TBD).

Every time period (TBD), the fusion will prepare and send a statistics update message to the Core Admin. The message will have the following structure:

|  |  |  |
| --- | --- | --- |
| # of reports | Data | CRC |
| Number of table lines to be reported in this packet | Terminal ID1DDMMYYYY hhmmss Rx Tx;Terminal ID2…. |  |

**Connection of the Fusion to the Modems Control protocol**

**Modems Control Protocol**

The modems control module is communicating with the modems card using standard UDP socket connection over Ethernet and a hiSky proprietary UDP based management protocol.The connection to each modem card includes three IP addresses – one for Tx, one for Rx and one for control. Each modem card supports 20 channels. The multiplexing of the channels is based on the UDP port – a port for each channel. For the MVP the HUB PP will only use one channel, so this requires 3 connections – one for Tx, one for Rx and one for control. The control (“PHY control”) is not part of this section and is described in another part of this document. So, for the following description, the actual implementation of Tx and Rx are done on different sockets.

*Comment: iDirect are using VLANs for the separation of data and control traffic. The exact details of this VLAN scheme are not known yet. It is not expected to have direct implications on the description in this section*.

The protocol includes Opcode for the transfer of user data packets and control packets. Each packet will have a unique 2 Bytes Sequencenumber. For each data packet, an ack message will be sent to approve it was received.A NACK packet is sent when the TX buffers are full. The protocol allows to aggregate more than one TS per packet. In the MVP will only one TS per packet will be tested.

The logic will be as follows:

1. A data packet is sent.
2. For each data packet that is sent and received correctly on the other side, the receiving side will send an ACK message.
3. An ACK packet carries the serial number of the ACKed packet.
4. The sending side will wait ack\_timeout to receive an ACK message.
5. If an ACK was received, the next data packet can be sent.
6. If an ACK message was not received after ack\_timeout, the sending side will retransmit the packet.
7. The retransmission will only repeat ack\_retransmit\_count.
8. If one side wants to signal a back pressure, it will send a NACK message. The NACK will carry the serial number of the last packet that was not processed.
9. When one side receives a NACK, it will resend the last packet ASAP.
10. The retransmission of NACKed packet will only repeat nack\_retransmit\_count.
11. All ACK and NACK packets will contain a field for buffer free apace indication. The buffer free space will be indicated in number of TSs.
12. ACK and NACK packets will be counted and periodically put into a log file on the HUB PP side for statistics.
13. If ack\_ retransmit\_count or nack\_retransmit\_count have been exceeded, the event will be logged.

Following is the packets descriptions:

1. Data packet:

This is regular data packet. The Seq # is a serial number of the packet for this protocol only. The preamble comes to separate between TSs in the same packet and its contents is TBD.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Seq # | Opcode | TS1 Preamble | TS1  Length | TS1 data | … | TSn  Preamble | TSn  Length | TSn  data |  |
| 2Bytes | 1 Byte  0 = Regular Data | 2 Byte | 2Bytes |  | … | 2 Byte | 2Bytes |  |  |



1. ACK Packet:

This packet is sent after each data packet that is received correctly.

|  |  |  |  |
| --- | --- | --- | --- |
| Seq # | Opcode | Buffer status | Serial number ACKed |
| 2Byte | 1 Byte  1 = Ack | 1 Byte  # of free TSs in buffer | 2B |

1. NACKPacket:

This packet will be sent if one side receives a packet for sending, but itcannotsend it.

|  |  |  |  |
| --- | --- | --- | --- |
| Seq # | Opcode | Buffer status | Serial number NACKed |
| 2Byte | 1 Byte  2 = NACK | 1 Byte  # of free TSs in buffer | 2B |

**Other ICD Description**

Service to Fusion ICD

Each service connects to the fusion using a UDP socket. The service will act as a client in this connection, and the fusion will be the server. Upon connection, the service will supply the fusion a unique service ID. The service ID will be given to the service by the Core admin, upon the creation of the service by the core admin. The same file will also indicate the port that the service will use as a source port when it connects to the fusion. The communication between the service and the fusion will be in UDP packets format, using the localhost IP 127.0.0.1 as both source and destination and the port numbers of the fusion server and the service client.

HUB Fusion to Terminal Fusion ICD

Core Admin to Fusion ICD

Fusion to Core Admin - Update message

Core admin to Fusion - Ping message

Core Admin to NMS ICD

HUB PP registration

Terminal Registration

Activity report

Core admin HUB to Core Admin Terminal

Reg. Req.

Reg. Reply

Errors report

Status report

Modem Control to HUB Modems ICD

Service to App Server ICD

**Technician GUI**

???

**Configuration Parameters (work in progress…)**

The following parameters will be defined in a local file or retrieved from the NMS.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Used by (Unit) | Suggested value | Description |
| App. Server IP address | Core Admin | TBD | The external app server IP address or hostname. May be different for different terminals and/or services. |
| App. Server port | Core Admin | TBD | The external app server UDP port. May be different for different terminals and/or services. |
| Service source port for the app server | Core Admin | TBD | Each service in the HUB PP will have a unique source port for the App server connection. |
| Service source port for the Fusion | Core Admin  & Fusion | TBD | Each service in the HUB PP will have a unique source port for the Fusion connection. |
| Ping time out | Core Admin | TBD | Time period to wait until the Core Admin sends a ping message to the terminal. |
| Inactive time out | Core Admin | TBD | Time period to wait from the time a ping was sent until the terminal is declared inactive. |
| Fusion server port | Fusion | TBD | The port to which the services and Core Admin are connecting with the fusion. |
| Initial\_short\_terminal\_ID | Core Admin | 1 | The first terminal short ID |
| Initial\_service\_source\_port\_fusion | Service and Fusion | TBD | Initial source port of the service to fusion socket connection |
|  |  |  |  |
| Initial\_service\_source\_port\_app | Service | TBD | Initial source port of the service to external app server socket connection |
| max\_terminals | Core Admin | <400 TBD | Maximum number of terminals per channel |