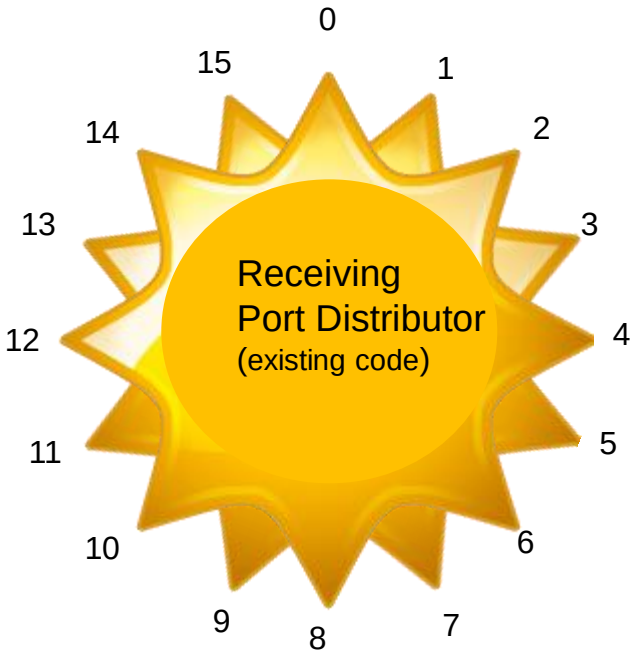
Packet Ports For Embedded Systems' Telemetry/Control



As Used By
Data Exchange with Remote (wired/wireless) Reprogramming
DXRP
(example Port-based App.)

Packet Ports For Embedded Systems' Telemetry/Control

Why Packet Ports?

- Simplify small embedded systems with use of IP-like Ports, some standardized.
- Applications can be transport method unaware.
- Independently developed Apps can coexist on different port numbers with different message formats
- On any port (except 0), message formats are known
- Built-in Message Distribution to correct port listener for each port
- Legacy applications use default port 0 and are unaffected by Packet Ports
- Easy to bridge the embedded systems' port packets to IP packets

Use Ports in small systems with small sized packets as used in embedded systems/telemetry?

- Yes, very small and efficient, small code size
- Destination ports, no source ports needed here. Bridge to IP by mapping port # and address to IP conventions
- Message content can indicate originator's (source) info.

Protocols, radios, range?

- Datagram services as in RadioHead (open source). With or without error correction
- Topologies: Star, Routed, Self-forming Mesh, or peer-to-peer
- Typ. < 100Kbps; Lower rates can yield 1+ Mile range w/dipole antenna, 100mW. More with routed/mesh protcol.
- Drivers for popular FM/FSK 2-way data packet radios, about $5.
- Inherent option for routing in the wireless itself, via static or mesh

Why not WiFi or Bluetooth?

- Many apps. need low cost low power consumption radios. Most here are sub-GHz ISM bands
- Plus good range that comes from narrow channel widths, lower bit rates (e.g., operate to -100dBm)
- WiFi/Bluetooth/BLE makes sense when 20MHz channel and high data rates but ~300 ft. range

Implemented?

- Yes, RadioHead's protocols and multi-radio drivers are well established. Supports port number passing in headers.
- DXRP (Data Exchange with Remote Programming) and OTAR is new, beta-tested with Packet Ports

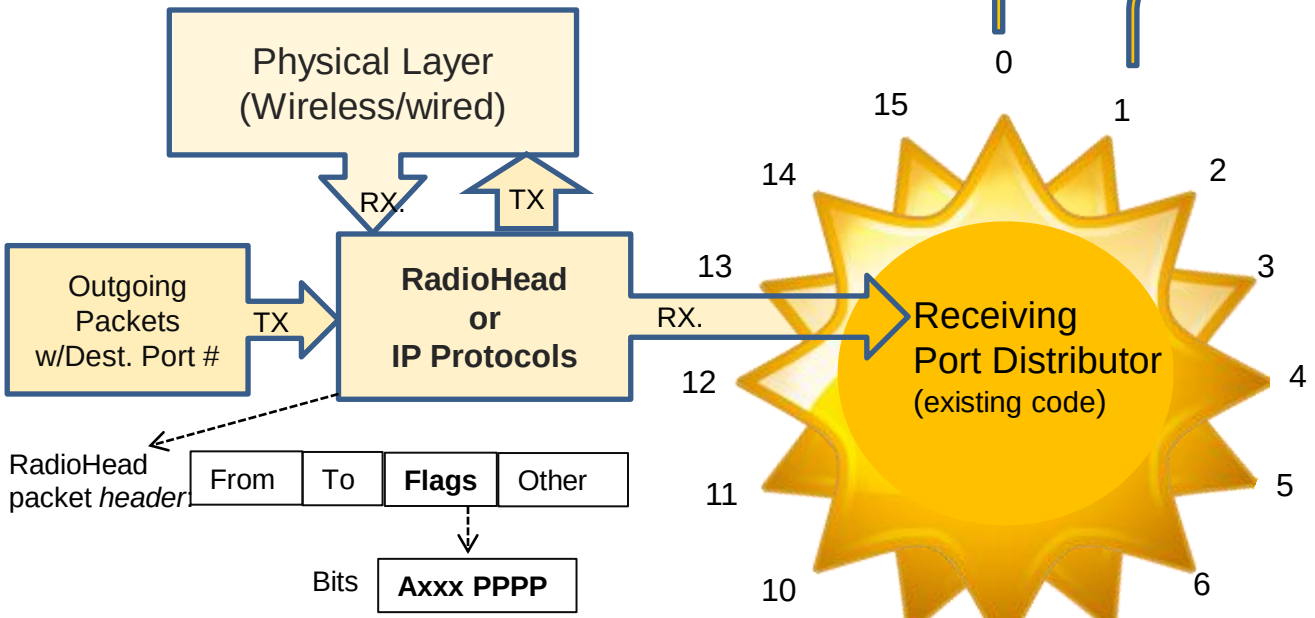
Packet Ports For Embedded Processors, Telemetry/Control

On Any One Port Number, Message Formats Are Known

(except for port 0)

Like IP, Some Reserved Ports Have Well-known Data Message Formats

Packet Ports and DXRP are implemented and tested as shown in later slide.



C Code **Service** for ***Legacy*** and non-port-based packets (will have port=0) Data Format is "any".

C Code **Service** for **DXRP** Formatted Messages. (DXRP = Data Exchange w/Remote Reprogramming) (Existing code)

Receiving Port Distributor calls a port's Service with data payload portion of a packet. Listener is transport-unaware as in IP systems' sockets.
Service process then returns a response packet to be sent the on same port #, or a NULL..

Listeners are defined at design-time or register themselves at run time.

Proposal:
Ports 1-6 Reserved For Utility/Universal Apps. Ports 7-15 are user-defined formats. Port range might extend to 32 or more.

C Code **Service** for Lighting Control (e.g.)

C Code **Service** for HVAC Heat Pump Control (e.g.)

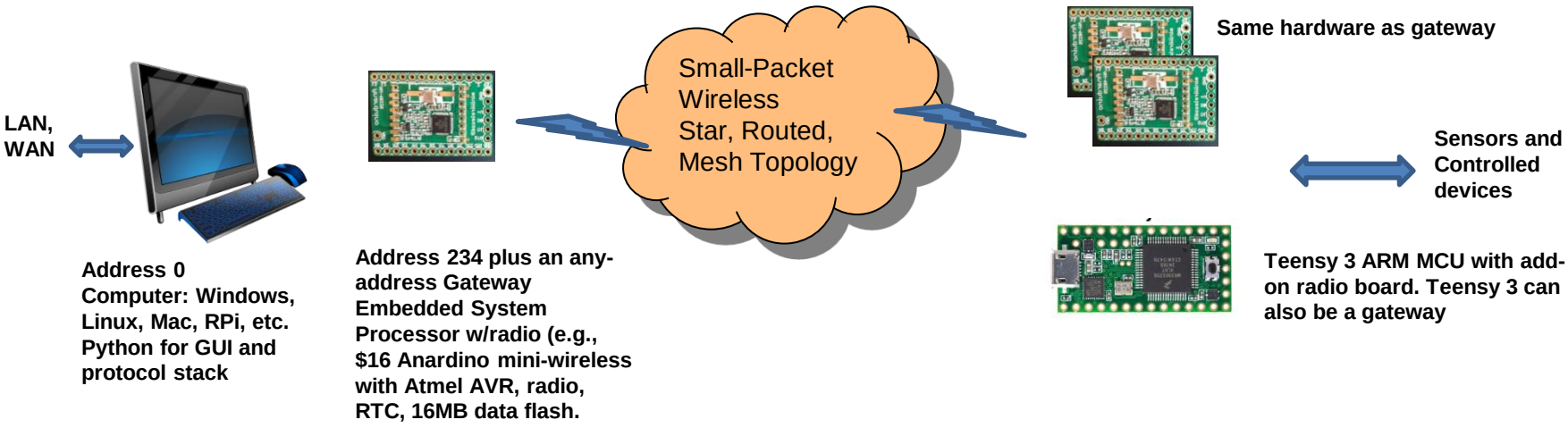
IPv4 packets to/from Radio Head:
Map RH port No to 16 bits
Copy RH payload verbatim
Translate RH address to IP Addr.

Example/Demo Ports App: DXRP

- Reliable bulk data exchanges (1-n hops, CRC for entire data set is passed)
- Tell recipient type of data, e.g,. Replacement program code vs. log data
- Discover network nodes' addresses in range of Gatway
- Authenticated remote command to reflash program just sent (OTAR)
- Get reports on nodes' code version, RSSI at node, etc., via "hello/ping" message
- Gateway node to make a Windows, Linux, Rpi, Mac *computer* an addressable network member
- Python (machine independent) program for *computer* DXRP control via graphical user interface

---

Demonstration DXRP Configuration - - Packet Ports and DXRP code on PC.
MCUs use Arduino-compatible libraries and C/C++.



Video Demo

Of Ports and DXRP as on prior slide

Port #TBD (1?) DXRP Messages Implemented and Tested as of 8/21/14

Note: The below omits the MAC layer ACKs and other link management messages

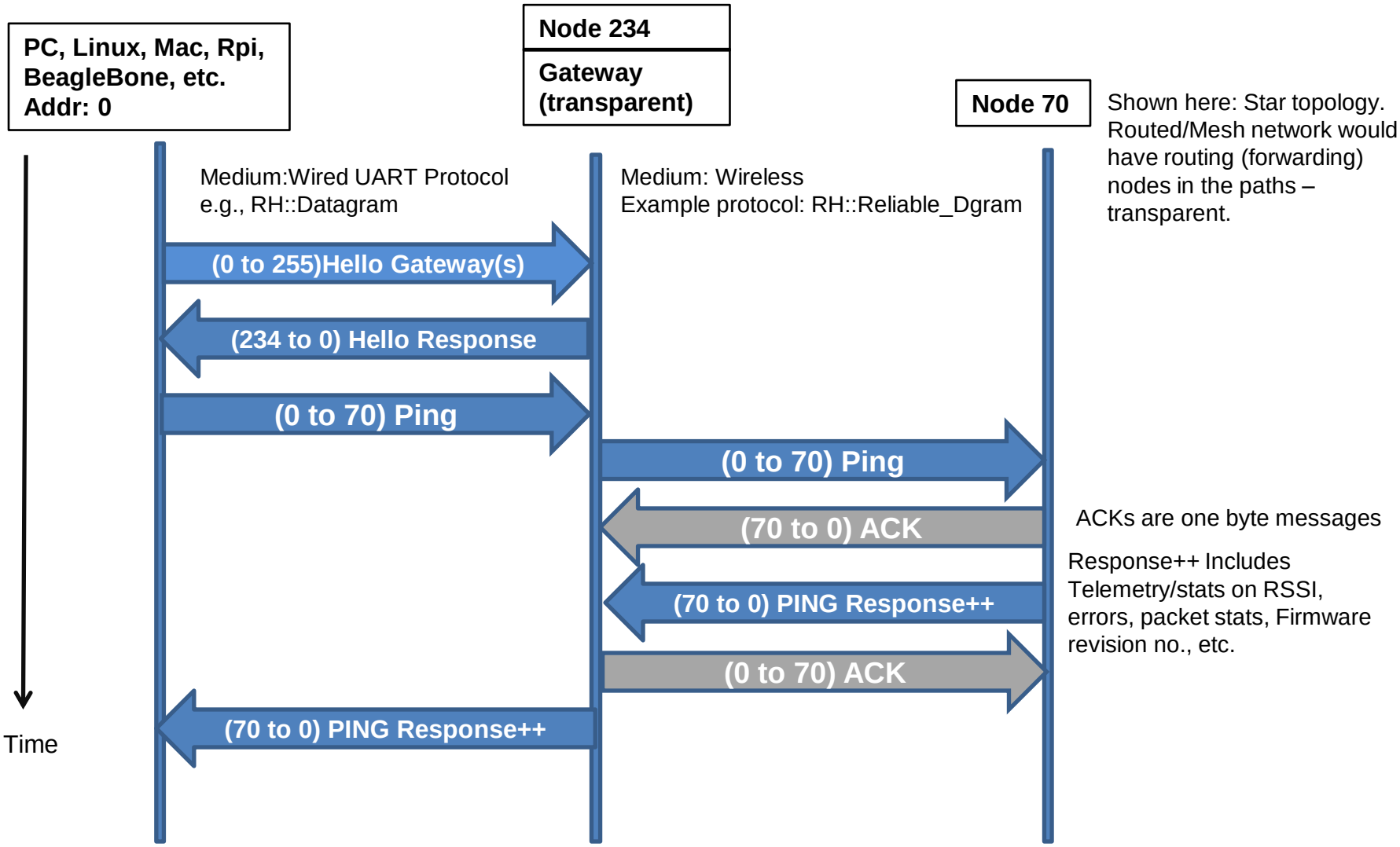
Message	Source	Destination	Explanation
Hello Gateway(s) (used at startup)	**Computer / Server**	**255 (all)**	**RH::Datagram. Medium:Typically point to point wired serial. Only gateways respond. Server discovers their addresses. Server can be embedded processor that uses same messages as a computer**
Hello / Ping	**Any**	**Any**	**RH::Reliable_Datagram. Medium/PHY: Typically wireless, sub-GHz, low cost, 100Kbps or less depending on desired range and link margin.**
Hello / Ping Response	**Addressee**	**Sender (typ. Computer, but any)**	**Protocol and PHY same as above. Response include telemetry on status, Firmware Version #, RSSI, etc.**
StoreTo (over the air reprogramming and data file send/receive)	**Server**	**Any**	**Protocol and PHY same as above. Asks if OK to store X bytes in storage place-code n, where n=0 if no-store, just test transfer. Tells receiver what the length and CRC of full entire data item shall be. Some StoreTo code n's are for extra program code variations or for data.**
ReadFrom			**Request and get data set previously stored at addressee. Not yet implemented.**
Accepted	**Addressee**	**Server**	**Response to StoreTo or Data Segment. Sent if received error free and Addressee can store and after completing storage work.**
Rejected	**Addressee**	**Server**	**Rejection of any message due to security denial/permissions or lack of resources such as storage space, or data error such as CRC fault.**
Data Segment	**Server**	**Any**	**Next data segment, typ. Max-sized for medium. Includes frame and user data CRC check codes for Addressee to use. Respond with Accepted or Rejected.**
Data Segment Done	**Server**	**Any**	**Tells addressee that all Data Segments have been sent. Addressee must do final processing including full-data CRC check, then respond with Accepted or Rejected message.**
Reboot/Restart	**Server**	**Any**	**Respond with Accepted or Rejected. If approved, addressee shall reboot/restart and if applicable, install newly received program code. Server delays and issues Hello/Ping and looks at response's Version #.**

Port Distributor and DXRP Testing status as of 8/21/14 (s. childress)

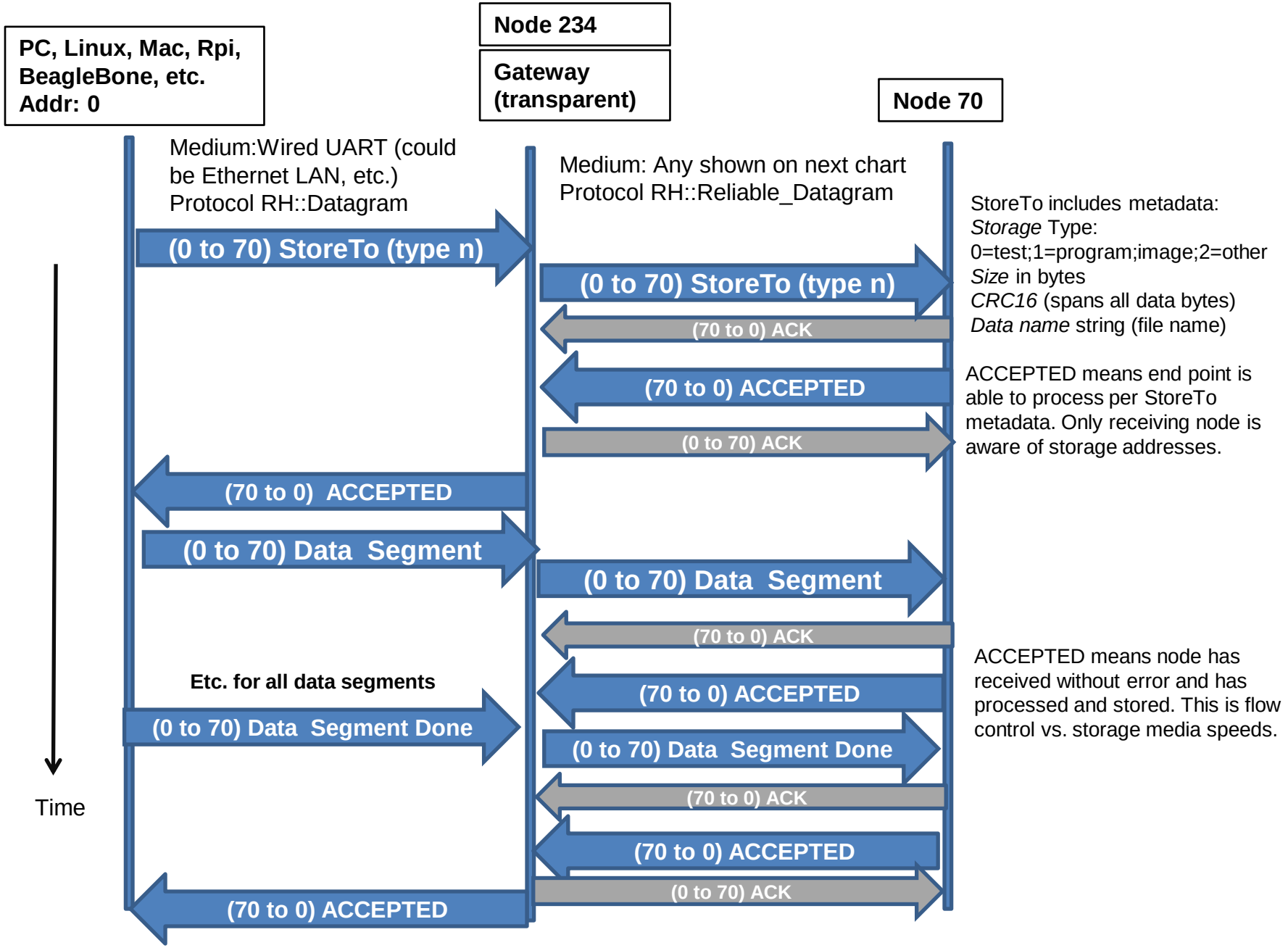
Note: The list below omits the MAC layer ACKs and other link management messages

Protocol Choice	Usage	Comment
RH::Driver (raw)	**Testing**	**Same frame/packet structure as below, but application program must populate the from/to addresses, send this raw packet, react accordingly.**
RH::Datagram	**With low error rate medium**	**Used for RS232, UART/UART, RS485, etc. where error checking at application layer is sufficient. No medium ACKs or retransmissions C++ base class is Driver**
RH::Reliable_Datagram	**With error prone medium**	**Used for wireless and some wired medium. Includes ACKs, retransmissions due to ACK timeout. Detects duplicate data and ACK packets based on packet sequence numbering (ID # in header). C++ base class is Datagram**
Below: Not yet tested with DXRP messages. Have been tested in other applications. Simple code change; testing more complex.		
RH::Routed	**Extend coverage in wireless, using "relay" routers**	**Static routing table entry add/delete messages are sent on the network. Each network node that routes has a table of in-range neighbor addresses. Nodes without routing tables are end points that do not forward or relay. C++ base class is Reliable_Datagram**
RH::Mesh	**As above, but for dynamic routing, e.g., due to node mobility or varying RF conditons**	**Simple form of self-forming, self-healing mesh routing (ad-hoc). Routing tables as in "Routed" above are created and updated automatically. Message latency can be longer in such a mesh. C++ base class is RH::Routed**

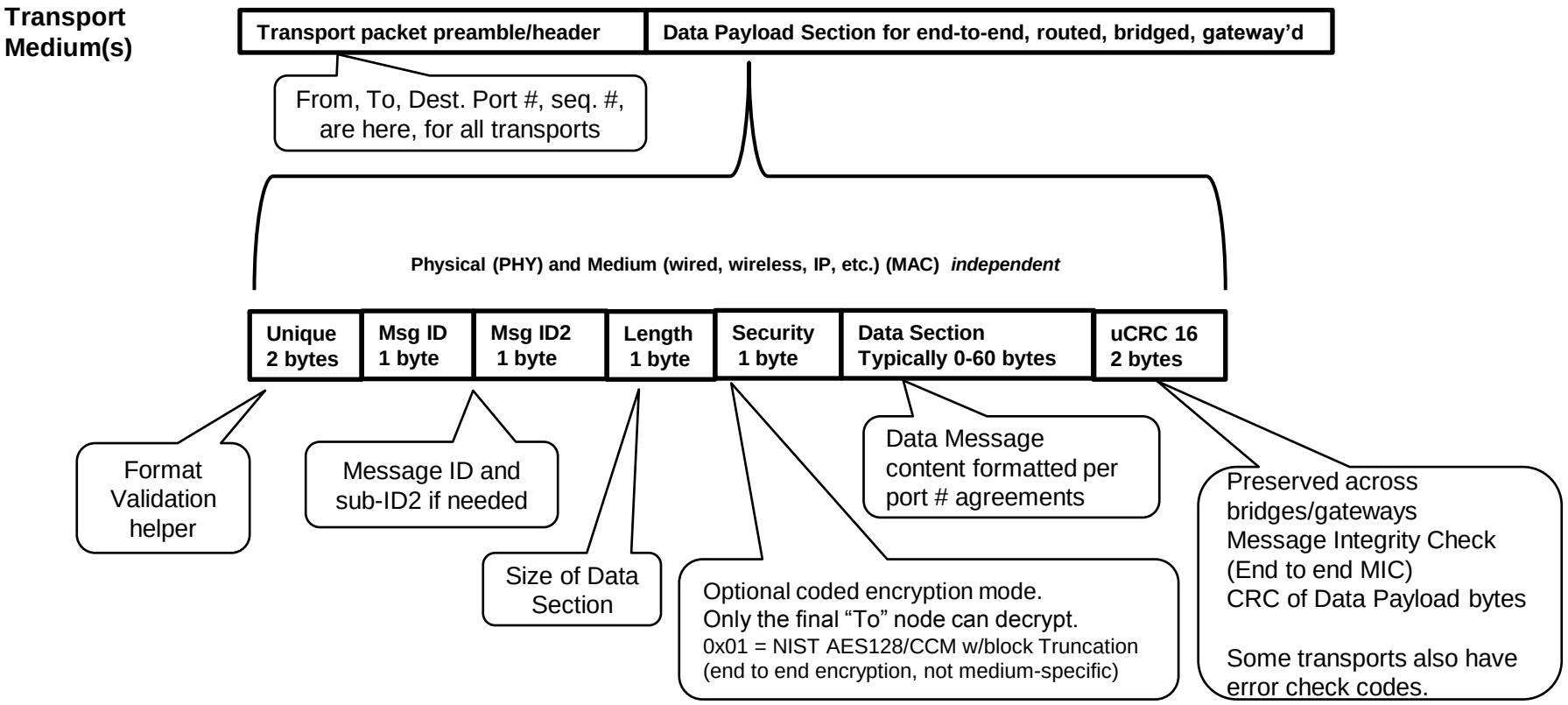
Message Ladder Diagram. DXRP Messages on Port TBD (1?)
Example Simple Messages: Computer to node via transparent gateway.
Startup and a Ping.



Message Ladder Diagram: Example of DXRP Bulk Data Transfer to end point – e.g., (OTA reprogramming)
Port TBD (1?) . Tested and working to include bootloader re-flashing at remote node.



Example payload (RadioHead or IP) data for DXRP use case on its port (1?). Other ports may adapt this format or define wholly new ones for that port # with a user system.



Unreliable datagrams can be used if messaging or that port # has application layer error correction.
Messages for Port #TBD (1?) as on prior pages can use unreliable datagrams in RadioHead bridge to UDP
DXRP's default is reliable datagram with error correction, presuming use of Wireless transport.