ALERT-2
Protocol Specification

Draft Version 0.1
January 8, 2008

Timothy J. Salo
Salo IT Solutions, Inc.
1313 5th Street SE
Minneapolis, MN 55414-4504
salo <at> saloits <dot> com
612-605-6896

This material is based upon work supported by the Department of Commerce under contract number DG133R07-CN-0175. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the Department of Commerce.

Contents

1. Overview

This document is the technical specification for the ALERT-2 protocols, a suite of next-generation wireless communication protocols for use in automated flood warning systems (AFWS). The ALERT-2 protocols will provided an enhanced alternative to the original ALERT and IFLOWS protocols (although many networks will continue to use these legacy protocols for the foreseeable future). These new wireless communications protocols will offer increased functionality and will facilitate the development of new capabilities for automated flood warning systems. They will use the new modem technology that has recently been developed by Blue Water Design LLC, but can easily be adapted to use other transmission media.

The protocols and other technologies specified here are intended to conform to and fulfill the requirements specified in a companion document, the "ALERT-2 Requirements Specification".

The primary audience of this document is software developers who wish to implement the ALERT-2 protocols. This document assumes that the reader has a basic understanding of network protocols and an exposure to protocol specifications. This is the first draft of the ALERT-2 Protocol Specification; some parts of this Specification require additional detail. It is being released now for review by a wider audience so that comments, suggestions, and corrections can be integrated into the next revision. As such, this initial draft should *not* be used as a guide for implementations. This document is available on the ALERT-2 project Web site (http://www.alert-2.com/). Please send any questions, comments, or suggestions you might have about this material to the author.

This document is a product of the *ALERT-2 Protocol Development* project, an SBIR Phase I contract awarded to Salo IT Solutions, Inc. (SaloITS) by the National Oceanic and Atmospheric Administration (NOAA). Of course, any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of NOAA or of the Department of Commerce.

2. ALERT-2 Terminology, Components, and Concepts

This section specifies terminology and describes components and concepts with which the ALERT-2 protocols are constructed.

2.1 Node

An ALERT-2 *network node,* or simply *node,* is a device that implements the ALERT-2 wireless communications protocols. A node uses the ALERT-2 protocols to communicate with other nodes.

It is sometimes useful to describe a node as playing one of three roles within an ALERT-2 network:

- **Remote Node**. The primary mission of a remote node is to collect and transmit sensor data; it is the source of the sensor data that the ALERT-2 network transports. It reads sensor data from locally attached hydrologic or other sensors and transmits those sensor data over the ALERT-2 network. Often, conserving energy is important for a remote node. Therefore, some of these nodes may enter a low-power state for extended periods of time. Remote nodes do not assist other nodes in transporting sensor data through the network. In fact, some remote nodes may not even be able to receive packets, but may only transmit packets.

- **Router**. A router helps other nodes transport their sensor data through the network. It receives packets from other nodes and transmits these packets towards the packets' destinations. Routers increase the geographic extent of a network beyond the direct transmission range of individual nodes. Most routers must be prepared to receive a packet from another node at any time. Some routers have sensors attached to them.

- **Base Station**. A base station is the destination for sensor data. Some base stations may also help manage the network, such as by enabling a systems administrator to configure a router or remote node over the ALERT-2 network.

The following terms commonly used when discussing low-power wireless networks may also be helpful:

- **Reduced-Function Device**. A Reduced-Function Device (RFD) is a network node that has severe resource constraints. These constraints typically include limited electrical energy, but often also include limited computational power and limited storage capacity. An RFD may conserve energy by entering a low-power state, often called a "sleep state" or "sleeping". In most environments, an RFD powers down the radio receiver while in a sleep state in order to conserve energy. As a result, a node is unable to receive packets while it is sleeping[1]. In ALERT-2 networks, many remote nodes can be described as reduced-function devices.

[1] Some systems, such as IEEE 802.16, enter an "idle" state, an energy-saving state in which the receiver is still active. When an 802.16 node is idle and receives a packet, it wakes the CPU to process the incoming packet. The ability of a node to receive packets while in a low-power state is not applicable to ALERT-2 networks.

- **Full-Function Device**. A Full-Function Device (FFD) is a network node that does not face severe resource constraints. Usually, an FFD operates continuously, and is prepared to receive a packet at any time. Typically, an FFD has more computational resources and storage capacity than do RFDs. In many network architectures, an FFD will perform functions for RFDs, thereby reducing the energy required by the RFDs. For example, in many protocols designed for low-power wireless networks, an FFD will hold a packet that is destined for an RFD until the RFD wakes up and is prepared to receive the packet. In ALERT-2 networks, routers and base stations are generally full-function devices.

2.2 Network

An ALERT-2 *network* is a collection of nodes that communicate with each other, perhaps indirectly, using the ALERT-2 protocols. In general[2], all of the nodes in a network are administered by a single organization. ALERT-2 networks generally do not communicate with each other (i.e., a node that is part of one network does not generally communicate with a node that is part of another network). Figure 1 illustrates the configuration of a typical ALERT-2 network.

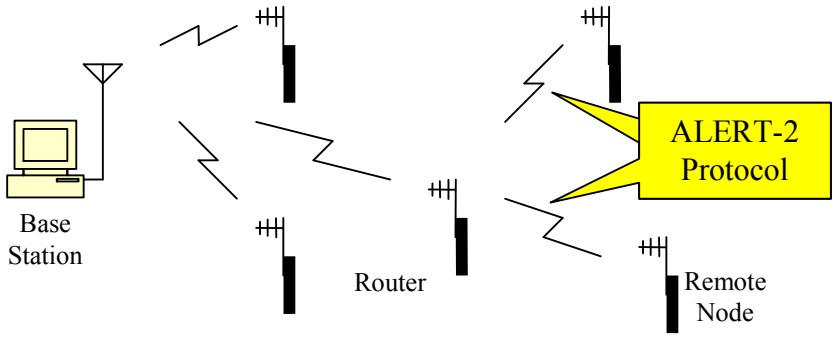


Figure 1. A Typical ALERT-2 Network

The principal mission of an ALERT-2 network is to transport sensor data from remote nodes to a base station.

2.3 Address

An ALERT-2 *address* is a 16-bit integer. An address, and generally only a single address, is assigned to each node. Each address within a network and within a geographic area must be unique. The ALERT-2 protocols use addresses to identify nodes.

A source address is an address that identifies the node from which a packet originated. Likewise, a destination address is an address that identifies the node to which a packet is being sent.

[2] The phrases "in general" and "generally" often indicate that the ALERT-2 protocols may not explicitly prohibit a particular activity, but that most users are unlikely to configure their network such that the specified activity occurs. This editorial technique is employed in order to simplify the specification of the ALERT-2 protocols.

The ALERT-2 protocols [generally] use only one type of address to a identify node. This contrasts with other network architectures, which may assign both link addresses and network addresses to each node. In the ALERT-2 protocols, the same address is used for both purposes.

This address assignment strategy differs from that used in the original ALERT protocol, in that in ALERT-2 networks addresses are assigned to nodes, rather than to individual sensors. As a result, the number of nodes in a geographic region is limited by the 16-bit address space (although the network address feature described below may reduce this number somewhat, at least in practice). Furthermore, the number of sensors that can be supported by one network or in one geographic area is effectively unlimited.

In the initial version of the ALERT-2 protocols, an address is assigned to a node as part of the manual configuration of that node.

In a future version of the ALERT-2 protocols, addresses may optionally be assigned dynamically. That is, a node may request at execution time (e.g., when the node is first powered up) that an address be assigned to it. The node transmits a request for an address, which includes a special identifier permanently assigned to that node. This permanent identifier may be a 48-bit IEEE MAC address or any other globally unique identifier. One node, typically a base station, will be responsible for processing all address requests. This node will have a database that maintains the mapping between permanent identifiers and corresponding ALERT-2 addresses. When this node receives a request for an address, it consults its database and informs the requesting node of the address that is assigned to it. This capability is roughly analogous to the Internet Dynamic Host Configuration Protocol (DHCP) although the analogy is not perfect (in part because an ALERT-2 network may span several broadcast domains).

2.4 Network Address

The ALERT-2 network administrators in a geographic area may mutually agree that 16-bit ALERT-2 addresses are divided into a *network address* and a *node address*. The network address part of the 16-bit address identifies the network to which a node belongs, while the node address part of the 16-bit address uniquely identifies the node within that network. For example, network administrators may decide that the upper four bits of addresses contain the network address and that the lower 12 bits of addresses contain the node address. This permits 15 networks to operate in the same geographic area, and each network can have as many as 4095 nodes. Each network is assigned a network address that is unique in the geographic area. Every network administrator ensures that the addresses that he or she uses have the network address set appropriately.

The benefit of dividing ALERT-2 addresses into a network address and a node address is that it allows networks in the same geographic area to be administered independently. That is, this feature eliminates the need for network administrators to coordinate the assignment of addresses with each other. The addresses assigned by one administrator are assured to be different than any addresses assigned by any other administrator because the network portion of the addresses assigned by different administrators is different. Additionally, the nodes in one network will generally be configured to ignore all packets that are associated with another network. A node

concludes that a packet is associated with another network if the network address of the packet's destination address does not match the network address of the node's address. This capability is roughly analogous to the Internet Classless Inter-Domain Routing (CIDR) technique, although the analogy is not exact.

2.5 Interface

An *interface* is the connection between an ALERT-2 node and a transmission medium. For example, a transceiver that operates at 169.xx MHz is an interface on the node to which it is connected. Another transceiver that operates at 171.xx MHz connected to the same node is a second interface on that node.

Conceptually, every ALERT-2 node has a *null interface*, an interface that discards all packets that are sent to it.

The purpose of the ALERT-2 interface concept is to simplify the specification of the ALERT-2 protocols. It is similar to an Internet interface, except that ALERT-2 addresses are assigned to nodes, whereas Internet addresses are assigned to interfaces.

2.6 Router

An ALERT-2 *router* is a node that *forwards* packets. Forwarding is the process in which a node transmits a packet that it received from another node in an effort to ensure that the packet reaches the node that is supposed to ultimately receive it.

The ALERT-2 router generally equivalent to what the original ALERT protocol called a repeater, although the behaviors of a router are more precisely specified.

2.7 Routing

Routing is the process by which a node determines how a packet should be forwarded. For example, a packet may be transmitted on the interface on which it was received, a packet may be transmitted on another interface, or a packet may be "transmitted" on the null interface (i.e., discarded).

A node determines how a packet should be forwarded by consulting its route table. A route table contains a number of entries, each of which includes the following information:

- Address – An address that is matched with the destination address of the packet being forwarded.

- Mask – A bit mask that specifies which bits of the destination address must be examined in order to determine whether this route table entry should be used; a mask allows a single route table entry to be applicable to many ALERT-2 addresses.

- Next Hop Interface – The interface on which packets that match this entry should be forwarded; a null interface indicates that the packet should be discarded.

- Next Hop Address – The address of the node to which packets that match this entry should be forwarded (using the Next Hop Interface).

In addition to specifying how packets should be forwarded, the route table specifies which packets are discarded. For example, all packets that are associated with other networks could be discarded (by specifying that the next hop interface on which these packets are to be transmitted is the null interface).

The route table in a remote node might contain only one entry, namely the address of the router to which it will transmit all packets. The size of the route table for a router may vary, depending on the topology of the network and the care with which addresses are assigned.

In the initial version of the ALERT-2 protocols, route tables are configured manually. In a future version of the ALERT-2 protocols, the route tables will be maintained dynamically using a routing protocol. A variant of the Internet routing protocol designed for mobile, ad hoc networks, the Optimized Link-State Routing (OLSR) protocol, is a candidate for use in ALERT-2 networks.

A more complete description of routing is deferred until after the ALERT-2 link and network protocols are described below.

2.8 Applications

A *software application*, or simply *application*, is software that executes on an ALERT-2 node and uses the ALERT-2 protocols to communicate with other applications. Several software applications may run on one node, perhaps a rain gauge reporting application that forwards rain gauge data to the base station, a file transfer application that transfers log files to the base station, and a network management application that reports the status of the remote node to the base station. From the perspective of the ALERT-2 protocols, the applications running on a node are independent, in the sense that the destination of a packet is a specific application on a specific node. For example, one packet could be sent to the sensor data collection application on node number 1313, while another packet could be sent to the network management application on the same node.

The purpose of an ALERT-2 network is to transport data between applications. Applications are, from the perspective of the ALERT-2 protocols, the sources and destinations of data.

The figure below illustrates how applications use the services provided by the ALERT-2 protocols to communicate with each other and how the ALERT-2 protocols provide services to several applications within a node.

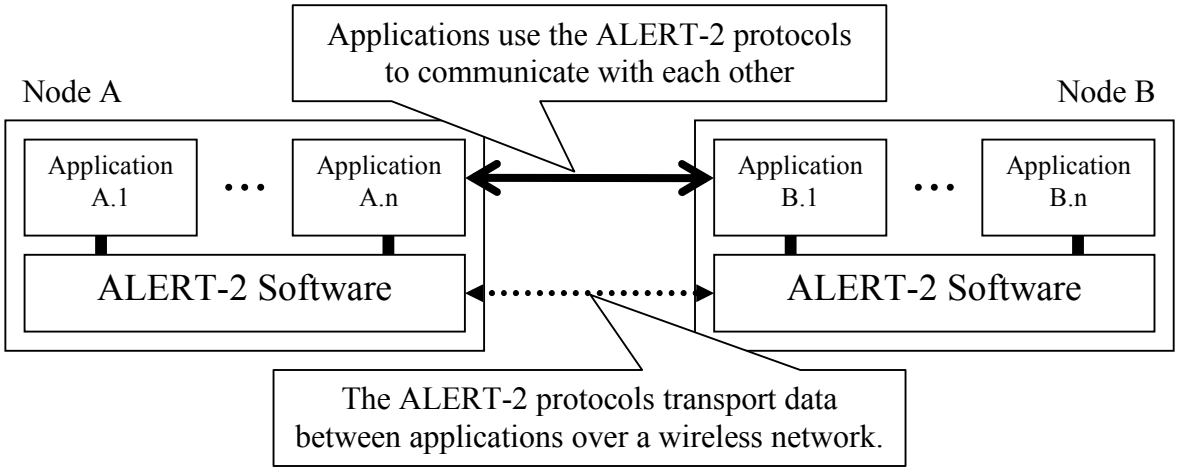


Figure 2. ALERT-2 Applications.

Applications offer several benefits:

- Applications make it easier to add new functionality to AFWS products or to modify the functionality of existing products. For example, adding a new network management application should not require any changes to an existing hydrologic sensor monitoring application. Likewise, modifications or enhancements made to a file system application (that might permit log files to be copied from a remote node to a base station) should not affect existing applications.

- Different applications may use different protocols, protocols that can be better matched to the needs of the individual applications than can a single, monolithic protocol. A sensor application, for instance, may transmit data to the base station using a best-effort datagram protocol (a protocol that transmits individual packets, but does not try to recover packets that are lost due to transmission errors, collisions, or congestion), while a file system application may use a protocol that ensures that all of the bytes of a log file are received correctly and in the right order.

2.9 Port Number

An eight-bit *port number*, or *port*, identifies a specific application within a node. Addresses identify the *node* from which a packet originated or to which the packet is destined. Port numbers identify the *application* (within the source node) that created the packet and the *application* (within the destination node) that is to receive the packet. ALERT-2 packets may contain a 16-bit source address, an eight-bit source port, a 16-bit destination address, and an eight-bit destination port. However, as describe in the following sections, some of these fields can be avoided in some packets.

2.10 Application Services

The ALERT-2 protocols provide three types of service to applications:

- **Best-Effort Datagram Service**. The *best-effort datagram service* transports through the network a single packet containing information provided by the application ("*application data*"). The ALERT-2 protocols make no special effort to recover packets that use this service that are lost due to transmission errors, collisions or congestion. This service is equivalent to that provided by the original ALERT protocol. It is analogous to the service provided by the Internet User Datagram Protocol (UDP).

- **Reliable Datagram Service**. The *reliable datagram service* transports a single packet containing application data. However, the ALERT-2 protocols take measures to ensure that a packet that uses this service is successfully received at by the destination application, even if the original copy of the packet is lost due to transmission errors, collisions or congestion.

- **Reliable Packet Stream Service**. The *reliable packet stream service* reliably delivers a sequence of packets containing application data to the destination application. All of the application data transmitted by the source application will be received by the destination application, and all of the application data will be received in the correct order. This service is equivalent to that provided by the Internet Transmission Control Protocol (TCP).

2.11 Protocol Layers

The ALERT-2 protocols are a collection of "layered" protocols. That is, each protocol performs a specific function, provides services to the protocol "above" it, and uses the services provided by the protocol "below" it. The use of a layered protocol architecture simplifies the introduction of new technology into or the addition of new functionality to ALERT-2 networks. For example, a new physical layer protocol could be added without affecting the other ALERT-2 protocols.

2.11.1 Physical-Layer Protocol

The ALERT-2 *physical-layer protocol* is responsible for transmitting bits over a radio frequency (RF) channel. The physical-layer protocol is provided by the 4800 bps modem with forward-error-correction (FEC) capabilities that is being developed by Blue Water Design LLC.

Additional ALERT-2 physical-layer protocols may be specified in the future. The layered protocol architecture used by the ALERT-2 protocol will allow new physical layer protocols to be specified without affecting the other ALERT-2 protocols.

2.11.2 Media Access Control (MAC) Protocol

The ALERT-2 *media access control (MAC)* protocol is responsible for determining which node can access (i.e., transmit on) the RF channel at a particular time. Most MAC protocols include features that reduce or eliminate "collisions", simultaneous transmissions by two or more nodes that interfere with each other and as a result prevent the transmitted packets from being successfully received.

The original ALERT protocol essentially had no MAC protocol. Each node transmitted data without regard to any external factors, such as whether another node might be transmitting at the same time. This strategy is known as the "pure ALOHA protocol". Unfortunately, the maximum throughput of the pure ALOHA protocol is about 18%, (i.e., the bandwidth used by the successfully received packets is at most about 18% of the total available bandwidth)[3]. As the rate at which packets are transmitted increases, collisions between packets increase; if the rate at which packets are transmitted is high enough, no packets are successfully received. This is undoubtedly the cause of the large number of lost packets that are experienced in ALERT networks during major rain events.

This initial version of the ALERT-2 protocol specification does not specify a MAC protocol. The creation of a MAC protocol is outside of the scope of the current contract and the available funds. The author intends to prepare a paper for the upcoming ALERT Users Group meeting that discusses the design, implementation, use, and benefits of time-slotted MAC protocols.

2.11.3 Link Protocol

The ALERT-2 *link protocol* is responsible for the orderly, error-free transmission of packets between two adjacent nodes. While most link protocols include mechanisms that detect and discard packets that are damaged by transmission errors, the ALERT-2 link protocol will rely upon the Blue Water Design modem to detect transmission errors and discard corrupted packets. The link protocol will provide two types of service: best effort and reliable. If a packet is transmitted using the reliable service, the link protocol will attempt to retransmit the packet, if it is lost due to transmission errors, collisions, or congestion. If the packet is transmitted using the best effort service, and if the packet is lost due to transmission errors, collisions, or congestion, the link protocol will make no effort to retransmit the packet and the packet is lost.

2.11.4 Network Protocol

The ALERT-2 network protocol is responsible for the end-to-end transmission of packets through a network, which may include one or more intermediate nodes (routers) that forward packets towards their destinations. The function of the ALERT-2 network protocol is roughly analogous to that of the Internet Protocol (IP) in the Internet protocol suite.

2.11.5 Transport Protocols

The ALERT-2 transport protocols are responsible for the end-to-end transmission of data. There are three ALERT-2 transport protocols: a best-effort datagram protocol, a reliable datagram protocol, and a reliable packet stream protocol. These transport protocols provide the three types of application services described above.

2.11.6 Application Protocols

The ALERT-2 application protocols are responsible for the end-to-end transport of application data. Each application may specify its own protocols. An application may specify an

[3] See, for example: Tanenbaum, Andrew S., *Computer Networks 4e*, Prentice Hall PTR, 2003, pp 251-254.

application protocol that uses a datagram service or an application protocol that uses the reliable packet stream service, or two protocols (i.e., a datagram protocol and a reliable packet stream protocol).

3. ALERT-2 Protocols

This section specifies the packet formats and protocols for the ALERT-2 protocols.

3.1 Physical-Layer Protocol

The ALERT-2 physical-layer protocol is provided by the Blue Water Design modem. Additional information will be provided or referenced as it becomes available.

3.2 Media Access Control (MAC) Protocol

This initial version of the ALERT-2 protocols does not specify a MAC protocol. Depending on the results of future investigations of the design and utility of a time-slotted MAC protocol, a future version of this specification may include a MAC protocol.

3.3 Link Protocol

The ALERT-2 link protocol is responsible for the orderly delivery of packets between two adjacent nodes. The fields of link-layer packets are specified in Table 1, although the precise ordering of the fields and the exact layout of the link-layer packet are not specified in this version of this document.

Table 1. ALERT-2 Link Protocol Packet Fields.

Field	Size	Function
Link Control Byte	1 Byte	Flags that control the behavior of this packet
Control/Data	2 bits	Specifies the service being provided by this packet 00 Link control information, such as link-level acknowledgements 10 Unacknowledged (best effort) data 11 Acknowledged (reliable) data
Link-Layer Destination Address Elided	1 bit	Indicates that the destination address field is not present in the link-layer header of this packet. The link-layer destination is assumed to be the receiving node.
Network-Layer Source Address Elided	1 bit	Indicates that the source address field is not present in the network-layer header of this packet. The link-layer source address is used as the network-layer source address
Network-Layer Destination Address Elided	1 bit	Indicates that the destination address field is not present in the network-layer header of this packet. The link-layer destination address is used as the network-layer destination address. If neither a link-layer destination address field nor a network-layer destination address is present, the packet is to be forwarded to a default address (which is likely to be a base station).

Table 1. ALERT-2 Link Protocol Packet Fields.

Field	Size	Function
Ports Elided	1 bit	Indicates that the source port field and destination port field are not present in the network-layer header. The source port and the destination port are assumed to be zero.
Reserved	2 bits	
Source Address	2 bytes	Address of the node that transmitted this packet. Note that this may not be the [network-layer] address of the original source of this packet
Destination Address	2 bytes	Address of the node that is to receive this packet. Note that this may not be the [network-layer] address of the ultimate destination of this packet. This field is not present if the "Link-Layer Destination Address Elided" bit is set.
Packet ID	1 byte	Identifier of the packet. For an acknowledged packet (Control/Data = 11) this value is used to acknowledge the packet. For an unacknowledged packet (Control/Data = 10) this value can be used by the receiver to track the number of packets from the transmitting node that have been lost. The transmitter should increment this field by one for each packet transmitted to a particular destination. Separate Packet ID spaces are used for acknowledged and unacknowledged packets. While the values of this field should be assigned sequentially, each ID value is acknowledged independently (unlike many protocols, in which an acknowledgement implicitly acknowledges all packets prior to the packet being acknowledged). Several bits of this field might be used for other purposes, should the need arise.
Data	n Bytes	Control information or a network-layer header and data, depending on the value of the Control/Data field. The maximum length of the Data field is a topic for further study. The length of the data field is determined from the length of the physical-layer data and the values in the Control Byte.

Topics that warrant additional investigation include:

- The necessity for a link-layer flow-control mechanism, such as a receive window.
- The desirability of a link-layer congestion-control mechanism, in addition to or instead of a link-layer flow-control mechanism.

3.3.1 Physical-Layer Adaptation Layers

Adaptation layers will be specified for each physical-layer protocol. These adaptation layers will describe how the fields of the link protocol are mapped onto the services provided by the

physical layer. An adaptation layer will be specified for the Blue Water Design modem as more information about the modem becomes available.

3.4 Network Protocol

The ALERT-2 network protocol is responsible for the end-to-end delivery of packets through a network. Table 2 describes the fields of network-layer packets.

Table 2. ALERT-2 Network Protocol Packet Fields.

Field	Size	Function
Network Control Byte	1 Byte	A control byte
Version	3 bits	A three-bit version number. This field should have a value of 1. Future versions of the ALERT-2 network protocol should advance this field by one.
TTL	3 bits	Time-to-live. This field should be set by the [network-layer] source node. The default value is 7. A node receiving this packet should decrement this value by one, and discard the packet if it reaches a value of zero. (The purpose of this field is to ensure that, in the event of a persistent routing loop, packets are eventually discarded, rather than looping forever.)
Protocol ID	2 bits	Identifies the transport protocol. 00 Best-effort datagram transport protocol 01 Reliable datagram transport protocol 11 Reliable packet stream transport protocol
Source Address	2 bytes	Address of the original source of the packet. This field is not present if the Network Layer Source Address Elided flag is set, in which case the source address from the link protocol header be used.
Destination Address	2 bytes	Address of the ultimate destination. This field is not present if the Network Layer Destination Address Elided flag is set, in which case the destination address from the link protocol header is used.
Source Port	1 byte	Port of the source application. This field is not present if the Ports Elided flag is set, in which case zero is used.
Destination Port	1 byte	Port of the destination application. This field is not present if the Ports Elided flag is set, in which case zero is used.
Data	n Bytes	Network-layer data (i.e., transport protocol header and transport-layer data)

3.5 Transport Protocols

The ALERT-2 protocols include three transport protocols: the best-effort datagram transport protocol, the reliable datagram transport protocol, and the reliable packet stream transport protocol.

3.5.1 Best-Effort Datagram Transport Protocol

The best-effort datagram transport protocol transports a single packet of data through the network. The unacknowledged link-layer service is used to transport the packets between nodes. No effort is made to detect or recover packets that are lost because of transmission errors, collisions, or congestion. In fact, this transport protocol has no header and provides no service beyond that provided by the lower-layer protocols. Figure 3 illustrates the structure of a best-effort datagram transport protocol packet.

Table 3. ALERT-2 Best-Effort Datagram Transport Protocol Packet Fields.

Field	Size	Function
Data	n Bytes	Application data

3.5.2 Reliable Datagram Transport Protocol

The reliable datagram transport protocol makes an effort to ensure that a single packet of data is reliably transported through the network. The acknowledged link-layer service is used to transport packets between nodes. End-to-end acknowledgements and retransmissions are used to ensure that the packet is successfully received by the destination. Figure 4 shows the fields in a reliable datagram transport protocol packet.

Table 4. ALERT-2 Reliable Datagram Transport Protocol Packet Fields.

Field	Size	Function
Reliable Datagram Control Byte	1 Byte	Controls acknowledgements
Data/Ack	1 bits	Set if this is an acknowledgement, clear if this is a data packet
Packet ID	7 bits	Identifier of the packet. This value is used to acknowledge the packet. The source node should increment this field by one for each packet transmitted to a particular destination. Alternatively, the source node can use a common space for all destinations, but with a slight risk of experiencing sequence number wrapping. Separate Packet ID spaces are used for the reliable datagram and reliable packet stream transport protocols.
Data	n Bytes	Application Data

The destination node sends an acknowledgement to the source node when a reliable datagram packet is received. The destination node discards the packet if a packet with the same sequence number has been received recently. The definition of "recently" is a topic for further study, as is the definition of a receive window.

The source node retransmits the packet, if an acknowledgement is not received within a retransmission time-out period. The value of the retransmission time-out value is a topic for further study, as is the maximum number of retransmissions and a retransmission back-off algorithm.

3.5.3 Reliable Packet Stream Transport Protocol

The reliable packet stream transport protocol ensures that all of the packets transmitted by the source node are received by the destination node, and that all of the packets are passed to the application in the correct order.

Table 5. ALERT-2 Reliable Packet Stream Transport Protocol Packet Fields.

Field	Size	Function
Reliable Packet Stream Control Byte	1 Byte	Controls operation of reliable packet stream transport protocol
Version	3 bits	A three-bit version number. This field should have a value of 1. Future versions of the ALERT-2 reliable packet stream transport protocol should advance this field by one.
SYN	1 bit	Used to establish a connection
FIN	1 bit	Used to clear a connection
ACK	1 bit	Indicates the acknowledgement field is valid
Reserved	2 bits	
Acknowledgement	1 Byte	Controls acknowledgements
Data/Ack	1 bits	Set if this is an acknowledgement, clear if this is a data packet
Packet ID	7 bits	Identifier of the packet. This value is used to acknowledge the packet. The source node must increment this field by one for each packet transmitted on a particular connection. (i.e., the four-tuple defined by [source address, source port, destination address, and destination port]). The sequence number space is unique to a connection. While the values of this field must be assigned sequentially, each ID value is acknowledged independently (unlike many protocols, in which an acknowledgement implicitly acknowledges all packets prior to the packet being acknowledged).
Data	n Bytes	Application Data

The destination node sends an acknowledgement to the source node when a reliable packet stream packet is received. The destination node discards the packet if a packet with the same sequence number has been received recently. The definition of "recently" is a topic for further study, as is the definition of a receive window.

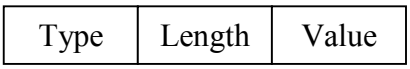
The source node retransmits the packet, if an acknowledgement is not received within a retransmission time-out period. The value of the retransmission time-out value is a topic for further study, as is the maximum number of retransmissions and a retransmission back-off algorithm. If the retransmission limit is reached, the connection is cleared.

The desirability of a congestion-control mechanism and a flow-control mechanism is the topic of further study.

3.6 Application Protocols

This version of the ALERT-2 Protocol Specification does not specify any application protocols. This specification recommends that application protocols be constructed as a sequence of Type/Length/Value (TLV) tuples.

A TLV is a highly flexible data structure that supports variable-format representation of data fields. TLVs have long been used to tag values in lower-level networking protocols. Using the TLV construct, each value is sent as a type/length/value tuple. The "Type" field is the tag that identifies the tuple, similar in function to HTML or XML tags. The "Length" field specifies the length of the value field, and the "Value" field is the actual data. Each message can be composed of one or more TLVs. Graphically, each TLV looks like:



Where:

> "Type" is a code that identifies the tuple. For example, the Type field might specify that the value field is a numeric value that represents the number of bucket tips that have been recorded since the device was reset. The Type field could be a fixed length (e.g., one or two bytes) or could be variable length (e.g., if bit 7 is set, another byte of Type follows).
>
> "Length" specifies the length of the Value portion of the TLV. Again, this could be a fixed length field (one or two bytes is pretty typical) or a variable length field.
>
> "Value" is a parameter associated with the Type code. A particular protocol may permit the Value field to contain one or more TLVs (i.e., the TLV data structure could be recursive).

TLVs are commonly used in Internet protocols, particularly lower-level protocols. They are used in BGP (an IP routing protocol) and RSVP (a quality-of-service reservation protocol).

A TLV-based protocol is easy to extend. Receivers are able to skip TLVs that they don't understand, and still find the next TLV (by looking at the Length field). RSVP goes a step farther, and classifies type codes as "if you don't understand this type code, just skip it" and "this type code is really important, so if you don't understand it, give up".

The RSVP TLV scheme is documented in RFC 2205, section 3.1.2, "Object Formats". Note that in RSVP the Length field is first, and the Type field is roughly a type field and a subtype field, (called "Class" and "Type" in RSVP). Section 3.10, "Future Compatibility", describes how a receiver should handle unknown Class value (or, in the language used here, a range of Type values).

TLVs enable a protocol to easily be extended without forcing the installed base to upgrade. Specifically, a device could be upgraded to send and receive new TLVs that support new functionality. Non-upgraded devices would continue to operate as before, simply ignoring the new TLVs. This ability for some, but not all, devices in a network to support new functionality provides a smooth upgrade path. Upgraded devices can be introduced into the network one at a time. Upgraded and non-upgraded would continue to communicate, but would not be able to use the new functionality. Pairs of upgraded devices would both communicate and take advantage of the new functionality. Old devices could either be upgraded over time, or could simply be left in place if the new functionality is not universally required. This smooth, incremental upgrade strategy enabled by TLVs contrasts with the "fork-lift" or "flash-cut" upgrades often required by fixed-format data structures.

The extensibility offered by TLVs is likely to be used in at least two circumstances:

- **New protocol features**. New features can be added to the protocol over time, while minimizing the effect on and incompatibility with the installed base.

- **Vendor-specific extensions**. TLVs-based extensions enable vendors to implement proprietary extensions in a fashion that is all but impossible with fixed-format data structures. Devices that do not support the vendor extensions can use the standard protocol to communicate with extended devices, while ignoring the vendor-proprietary extensions. Of course, it can reasonably be argued that the ALERT-2 Protocol Specification ought to assert that any device that *requires* the use of vendor-proprietary extensions does not conform to the standard.

TLVs provide a means by which a collection of readings from different sensors can easily and efficiently be combined into one packet (because the type of each sensor value is specified by the Type field).

TLVs offer numerous other advantages:

- **Transmission of only relevant fields**. TLVs enable a sender to transmit only the relevant data fields, rather than all of a fixed-format data structure. In cases where many fields are optional, where many fields are required only some of the time, or where some fields are required only occasionally, TLVs can reduce the number of bits transmitted. (TLVs may

increase the number of bits transmitted, if most fields are required most of the time. However, this cost must be traded off against the benefit of an easily extended protocol.) Note that contrary to what some claim, the use of TLVs does not imply that every field is optional. A standard can specify that certain TLVs are required in particular messages. The receiver can very efficiently verify that the required TLVs are present using bit fields and logical operations.

- **Improved software engineering**. Processing TLV-based messages is typically simpler (i.e., requires fewer branches in the source code) than processing fixed-format data structures. Typically, TLV processing code will be composed of a small, table-driven TLV parser and small procedures that process each of the TLV types. While in theory, fixed-format data structures could be processed by table-driven parsers, in practice most implementations are simply ad hoc, straight-line code.