

Specifications for
Autonomous Decentralized Protocol
R 3.0

MSTC/JOP 1101 (1999 September 30)

Distributed Manufacturing Architecture Committee
Japan FA Open Systems Promotion Group

MANUFACTURING SCIENCE & TECHNOLOGY CENTER

MANUFACTURING SCIENCE & TECHNOLOGY CENTER

Japan FA Open Systems Promotion Group
Distributed Manufacturing Architecture Committee

The protocol specifications defined in these specifications include patents, which MSTC is licensed for standard licenses from third parties without charge. The readers of these specifications must conform to the notes in the foreword to the body of the specifications.

Contents

Foreword	1
Introduction	2
1 Scope.....	2
2 Requirements for Protocol.....	3
2.1 Required Layer	3
2.2 Required Communication Protocol.....	3
3 Terms and definitions	4
3.1 ADP.....	4
3.2 Devices	4
3.3 LAN Segment.....	4
3.4 Logical Node.....	5
3.5 Data Field (DF).....	5
3.5.1 Local Data Field	6
3.5.2 Remote Data Field	6
3.6 Domain	7
3.7 Multi-cast Group.....	8
3.8 Broadcast	8
3.9 Multi-cast Communication.....	8
3.10 Peer-to-Peer Communication	8
3.11 Alive Signal.....	9
3.12 Fault Information	9
3.13 Application Program (AP).....	9
3.14 Protocol Data Unit (PDU)	9
3.15 Message	9
3.16 Transaction	10
3.16.1 User Transaction	10
3.16.2 System Transaction	10
3.17 Implementor.....	11
3.18 User	11
4 Protocol Functions	11
4.1 Class Base-1 (Multi-cast Communication)	11
4.1.1 Function.....	11
4.1.2 Managing Multi-cast Group	12
4.2 Class Base-2 (Transmitting Alive Signal).....	14
4.2.1 Function.....	14
4.2.2 Types of Transmitting Alive Signals.....	15
4.2.3 Alive/dead Judgement with Alive Signals	16
4.2.4 Assigning UDP Port Numbers.....	17
4.3 Class Opt-2-a (Transmitting Fault Information).....	19
4.3.1 Function.....	19

4.3.2	Fault Information.....	19
4.4	Class-Opt-3 (Peer-to-Peer Communication)	20
4.4.1	Function.....	20
4.4.2	Managing TCP Connection for Peer-to-Peer Communication.....	21
4.5	Test Support.....	23
4.5.1	Function.....	23
4.5.2	Message Modes.....	23
4.5.3	Node modes.....	24
4.5.4	Message Transmitting/Receiving Control.....	24
4.5.6	Assigning Multi-cast Receiving Ports	24
4.5.6	Operations of Test Support.....	25
4.6	Controlling Message Priorities	26
4.6.1	Function.....	26
4.6.2	Message Priority Levels.....	26
4.6.3	Prioritizing Messages	26
4.7	Sequential Numbering for Message Management	27
4.7.1	Function.....	27
4.7.2	Message Transmission Sequential Number	27
4.7.3	Versioning of Message Transmission Sequential Number	27
4.7.4	Managing Message Sequential Numbers	29
4.8	Dividing and Combining Message	29
4.8.1	Function.....	29
4.8.2	Information for Dividing and Combining Message	29
4.9	Associative Array Protocol.....	30
4.9.1	Message structure	30
4.9.2	Structure representation	31
4.9.3	Encoding example.....	32
5	PDU Structure and PDU Encoding.....	34
5.1	PDU Structure	34
5.2	Class Base-1 PDU (for Multicast Communication)	36
5.3	Class Base-2 PDU (for Alive Signals)	37
5.4	Class Opt-2-a (fault information)	39
5.5	Class Opt-1 PDU (for Peer-to-Peer Communication)	41
6	Conformance.....	42
6.1	Requirements for conformance.....	42
	Appendix A (Informative) TCD for System.....	43
	Appendix B Vendor Code List.....	44
	Appendix C (informative) Notes on Implementation.....	45
C.1	Message length	45
C.2	TCP connection management.....	45
C.2.1	Supplementary control information for connection	45
C.2.2	Fault detection for peer-to-peer communication path	45
C.2.3	Other notes	46

C.3	Message priority control.....	46
C.4	TCD Access Control.....	47
C.5	Logging statistic information for state change in node.....	47
C.6	Extensibility.....	48
Appendix D (informative)	Procedure	49
D.1	Transmitting/Receiving multicast communication	49
D.1.1	Transmitting	49
D.1.2	Receiving	49
D.2	Peer-to-peer transmission and reception.....	50
D.2.1	Transmitting	50
D.2.2	Receiving	51
D.3	Transmission number and version number management	52
D.3.1	For multicast communication.....	52
D.3.2	For peer-to-peer communication	55
D.4	Dividing and assembling messages.....	57
D.4.1	Dividing and assembling messages	57
D.4.2	Header information for fragmentation and assembly	58
D.4.3	Message fragmentation algorithm.....	59
D.4.4	Message assembling algorithm.....	60
D.5	Node availability monitoring.....	64
D.6	Fault information transmission	65
Appendix E (informative)	Sample Implementation.....	66
E.1	Byte order problem.....	66
E.2	Alignment problem.....	68
E.3	Alive signal.....	69
E.3.1	Conditions.....	69
E.3.2	Alive signal message setting values.....	69
E.4	Creating Fault Information.....	71
Appendix F (informative)	Example.....	72
F.1	Displaying node status	72
F.2	Displaying fault information.....	73
F.3	Specifications on error messages for system monitoring tools	74
Appendix G (informative)	Purpose of Autonomous Distributed System.....	75
G.1	Purpose of autonomous decentralized system	75
G.2	Features of autonomous decentralized system.....	76
Appendix H (informative)	Purposes of Associative Array.....	77
H.1	Features	77
H.2	Concept of This Implementation Draft.....	78

Specifications for Autonomous Decentralized Protocol

MSTC/JOP 1101 (1999 September 30)

Foreword

These specifications define the functionality and protocol of the autonomous decentralized interface planned by the Distributed Manufacturing Architecture Committee of the Japan FA Open Systems Promotion Group under the Manufacturing Science and Technology Center (hereafter called the Supplier).

Note that the protocol specifications defined in these specifications include patents, which the Supplier is licensed for standard licenses from third parties without charge. The users of the specifications are licensed for standard licenses regarding to the pertinent patents only when manufacturing or marketing their products conforming to the regulations defined under the specifications. The standard licenses shall not exist initially in relation to the users concerned with the specifications when they insist on their rights based on the patent related to the target products to the third party licensed for the standard licenses

The Supplier will provide the technical support on the specifications on its own authority, however, third parties introduced by the Supplier may provide technical support with extra costs.

The users of the specifications must observe the following restrictions:

- The users of the specifications cannot modify their contents except when the Supplier approves the modification.
- The users of the specifications can use them without charge to manufacture, create, use or market hardware or software products (hereafter called Target Products) provided they observe the functional specifications in the specifications and intend to use them on Ethernet.
- It is the sole responsibility of the manufacturer to operate a product that it has created based on the specifications, and the Supplier will undertake no responsibility.

Introduction

Many a traditional communication protocol standard is based on the peer-to-peer communication. The peer-to-peer communication is easy to recognize delivery, which has many advantages when providing a communicating way with high liability. On the other hand, it is accompanied with various difficulties in case of (i) monitoring by other nodes except the transmitter node and the receiver node, (ii) backup when a receiver node goes down, (iii) modifying the destination node address information in the transmitter nodes when the system is modified or expanded, and so forth. Especially, in case of decentralized systems such as manufacturing automation systems, they bring a serious problem.

The autonomous decentralized system, based on the broadcast communication, is such a system that works with each node autonomously choosing information to function as a whole. It is characteristic of the broadcast communication with no specific correspondent that the above mentioned problems do not occur.

The autonomous decentralized protocol is defined as a communication standard that should be used in such a system.

It uses exactly TCP, UDP/IP and Ethernet diffused widely as open communication networks and prescribes the items requested to the autonomous decentralized system on them.

1 Scope

The specifications define the following items about the communication protocol (called Autonomous Decentralized Protocol) based on the concept of autonomous decentralization.

- (1) ADP whose lower layer is TCP, UDP/IP or Ethernet
- (2) Functionality of the protocol
- (3) Encoding per protocol data
- (4) Functional restrictions required by the implementation applicable to the specifications

2 Requirements for Protocol

2.1 Required Layer

Figure 1 shows Layer required for implementing the functionality of Autonomous Decentralized Protocol defined by the specifications.

Application Programs
Autonomous Decentralized Protocol defined by the specifications
TCP/IP, UDP/IP
Ethernet based on IEEE 802.3

Figure 1 Required Layer

2.2 Required Communication Protocol

Before installing the ADP on a device, make sure that an Internet protocol (IP) conforming to any of the RFCs listed in table 1 has been installed on the device.

Table 1 Required RFCs

OSI layer	Protocol	Compliance target (RFC number)	Compliance level	Remarks
Physical layer		IEEE 802.3 compliance	Mandatory	
Data-link layer	Ethernet frame	RFC 894	Mandatory	
	ARP	RFC 826	Mandatory	
Network layer	IP	RFC 791	Mandatory	
	ICMP	RFC 792	Optional	
	For broadcast	RFC 919, RFC 922	Mandatory	
	For sub-network	RFC 950	Recommended	*1, *2
Transport layer	UDP	RFC 788	Mandatory	
	TCP	RFC 793, 761, 675	Optional	*3

NOTE:

*1: For devices not supporting the sub-networking, the system constraints must be spelled out.

*2: For any device not supporting the sub-networking, the operating manual must spell out the constraints as follows:

"The product does not support the sub-networking. Any system using the product cannot be built into a system using sub-networking."

*3: Mandatory for supporting peer-to-peer communication

3 Terms and definitions

3.1 ADP

ADP (Autonomous Decentralized Protocol) is the communication protocol that is defined by the specific actions.

3.2 Devices

Devices refer to the system components connected to a LAN, including physical controllers, calculators or controlling equipment.

3.3 LAN Segment

A LAN Segment refers to the extent of more than one LAN component connected through repeaters and/or bridges where one Ethernet frame can be transferred, or a LAN extent that is identified with one IP address.

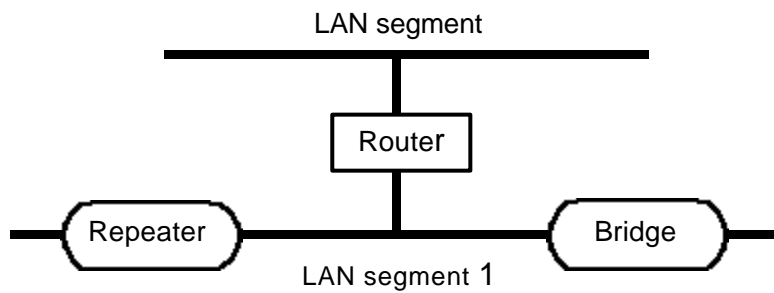


Figure 2 LAN segment

3.4 Logical Node

Logical Nodes refer to devices belonging to a data field. Within a data field, a number uniquely identifies a logical node is called a Logical Node Number (LNN). Unique LNN is assigned to each of devices in the range of 1 to 4095. Logical node number 0 is reserved for communication within the current node.

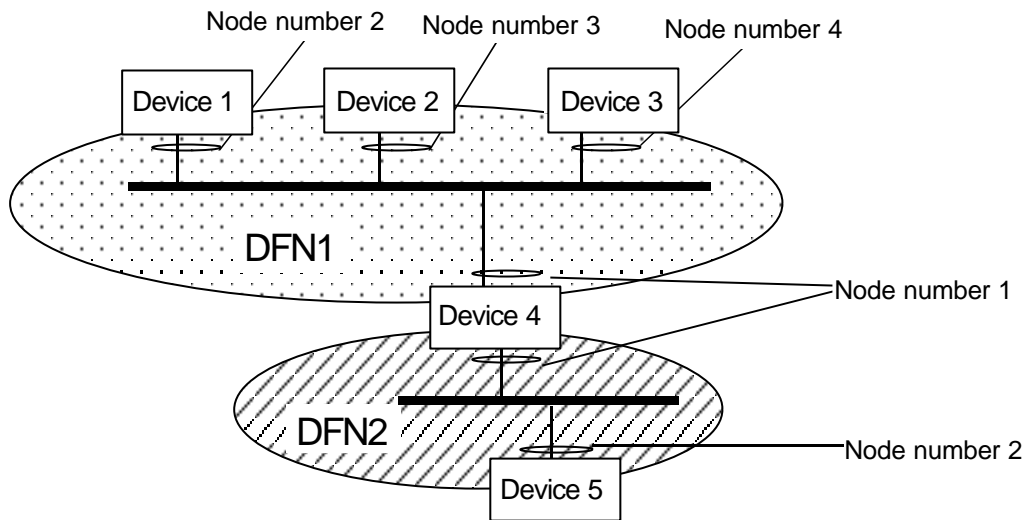


Figure 3 Logical node and logical node number

3.5 Data Field (DF)

DF is an area where ADP messages consisting of information of specific characteristics can flow. Communication using ADP is available between nodes belonging to one data field. This means that any device in a system must belong to one or more data field. One data field is configured per network address or sub-network address of an IP address. Within a decentralized system, a number uniquely identifies a data field is called an Data Field Number (DFN). Unique DFN is assigned to each of data fields within the system, in the range of 1 to 255. Data field number 0 is reserved for communication within the current node.

NOTES:

- No data field can be set up across multiple LAN segments. Every LAN segment must be assigned with DFN.
- One LAN segment may contain more than one data field.

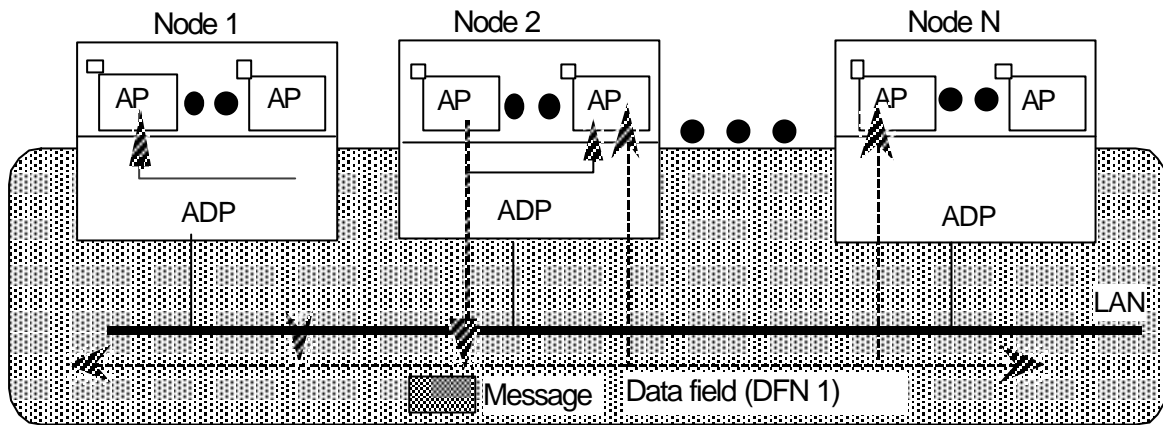


Figure 4 Data field and data field number

3.5.1 Local Data Field

A local data field is a data field where the current node is directly connected.

3.5.2 Remote Data Field

A remote data field is a data field where the current node and other nodes are connected through routers or gateways. For example, in Figure 5, data fields 1 and 2 are local to node 1 since they are connected directly to node 1, while data fields 3 and 4 are remote to node 1.

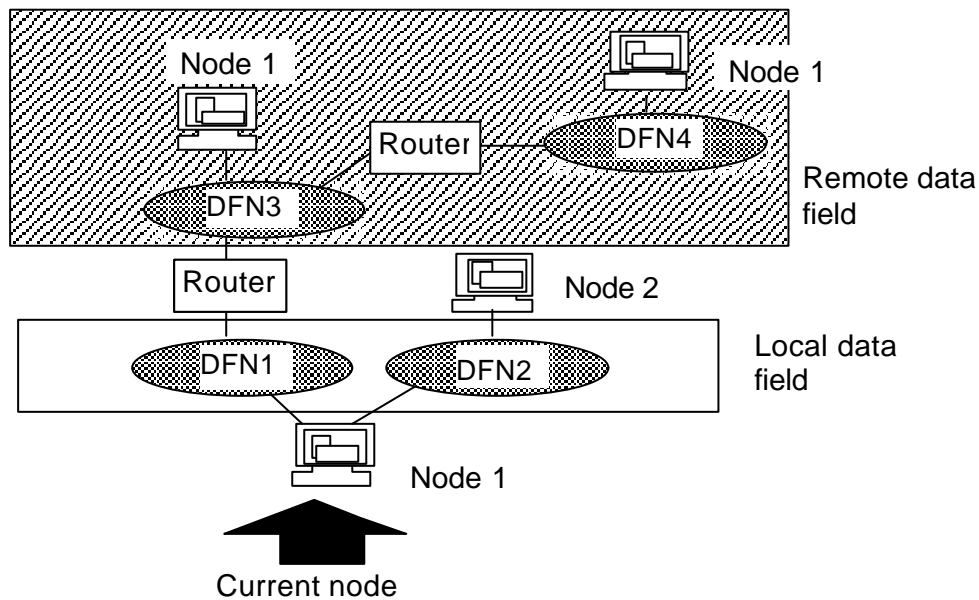


Figure 5 Local data field and remote data field

3.6 Domain

A domain refers to an upper concept of data fields, is a group of data fields. Generally each domain is defined as a local site, and connected with other domains through wide area communication networks.

Within a system, a number uniquely identifies a domain is called a Domain Number (DNN). Unique DNN is assigned to each of domains in the system in the range of 1 to 64. If domain number 0 was specified (i.e., a domain number was omitted), the system assumes the same domain has been specified. DNN 0 is reserved for the current DNN.

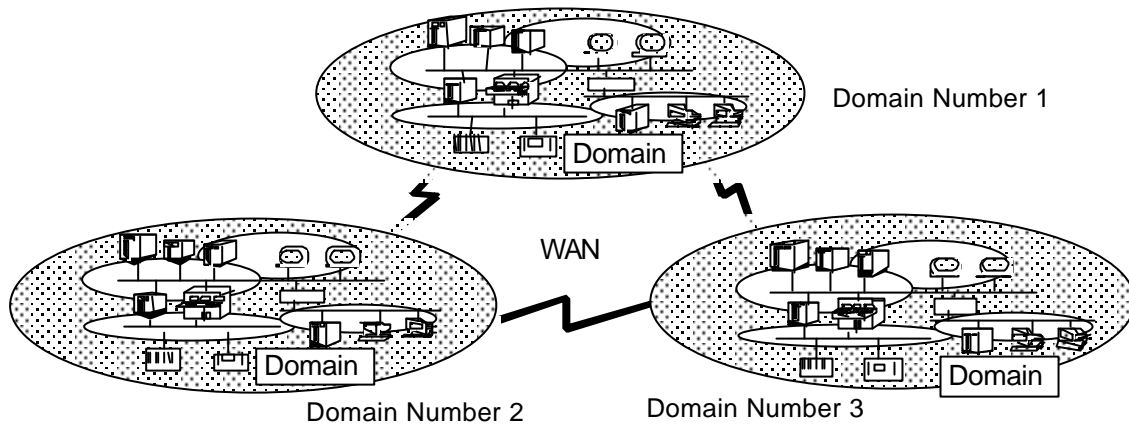


Figure 6 Domain and domain number

3.7 Multi-cast Group

A multi-cast group refers to a group of nodes set up within a data field. A multi-cast group must be specified to perform multi-cast transmission. Any node must belong to a multi-cast group to be used for multi-cast reception. One data field can contain more than one multi-cast group. Each node can be added to more than one multi-cast group as far as the groups are within a data field to which the node belongs. Multi-cast Group Number(MGN) is a number uniquely identifies a multi-cast group within a data field. Unique MGN is assigned to each of multi-cast groups in the data field in the range of 1 to 255. MGN 0 is reserved for the system to send/receive alive signals.

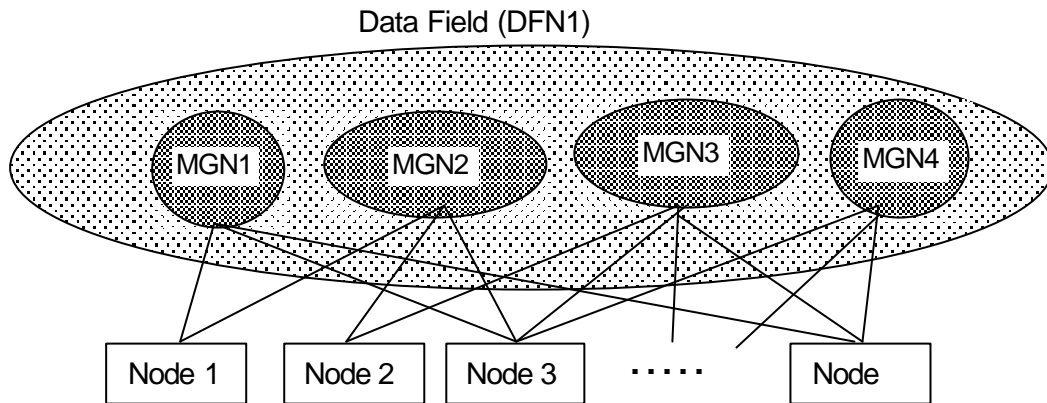


Figure 7 Multi-cast group and multi-cast group number

3.8 Broadcast

A broadcast refers to simultaneous transmission to all the nodes, where all the nodes receive the transmitted message.

3.9 Multi-cast Communication

The multi-cast communication is one of the transmission methods defined by the ADP, which communicates based on broadcasting through the UDP protocol. In this communication, one and the same message is sent to more than one node belonging to the multi-cast group that once requested the system for transmission.

3.10 Peer-to-Peer Communication

The peer-to-peer communication is one of the transmission methods defined by the ADP, which communicates based on the TCP. Before transmission, a peer-to-peer communication path (TCP connection) is established between both the end nodes. When AP requests for transmission, a communication message is sent to the specified target node.

3.11 Alive Signal

An alive signal refers to a message broadcasts to all other nodes in the same data field periodically to notify all the others in the field that it is alive.

3.12 Fault Information

Fault information refers to a message that a node periodically broadcasts to all the other nodes within the same data field to notify all the others in the field of the information on a fault occurred on the node.

3.13 Application Program (AP)

AP refers to an upper layer program used for application processing, including user processes and applications.

3.14 Protocol Data Unit (PDU)

PDU refers to a data unit passed to ADP from a lower layer and vice versa. It consists of a ADP header and data sections.

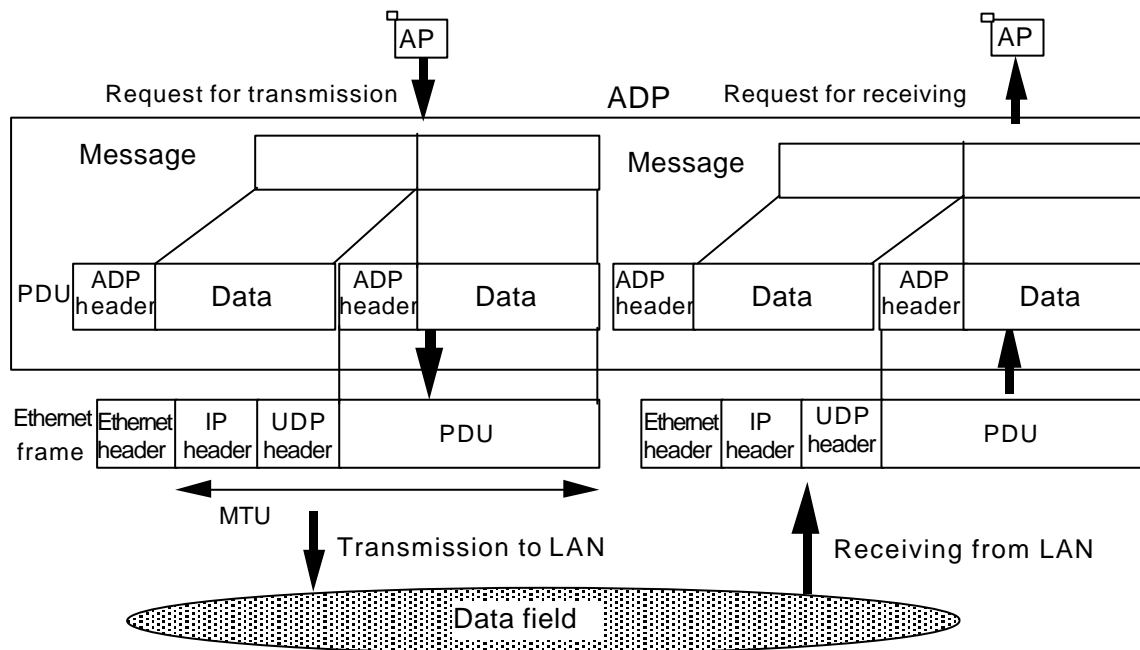


Figure 8 Message and PDU

3.15 Message

A message refers to the basic data unit passed to ADP from an upper layer and vice versa.

3.16 Transaction

A transaction refers to a process within AP or system. A code uniquely identifies a transaction is called a Transaction Code (TCD). AP specifies TCD before transmitting a message, or receives the information necessary for a UP process (transaction). The ADP includes TCD in the ADP header for communication. A unique TCD must be defined for each data field. There are two transaction types, the user and system transactions. The following ranges are allocated to the TCDs for users and for system:

TCD for users: 1 to 59999
 TCD for system: 60000 to 65534

3.16.1 User Transaction

A user transaction refers to a transaction within AP, application processing. A code identifying the associated user transaction (TCD for users) is assigned to every application process performed by a user.

3.16.2 System Transaction

A system transaction refers to a transaction generated when ADP detects any event (e.g., fault status or internal status change) and is used as a notice to AP. The codes identifying system transactions (TCDs for system) are used by implementors of various protocol functions.

AP must send/receive any message with TCD for the system independently of the node mode (Online or Test mode) or I/O control. Attached document A lists the relation between the system transactions and system TCDs.

The specifications only define the equation: $TCD=60003$ (alive signal)

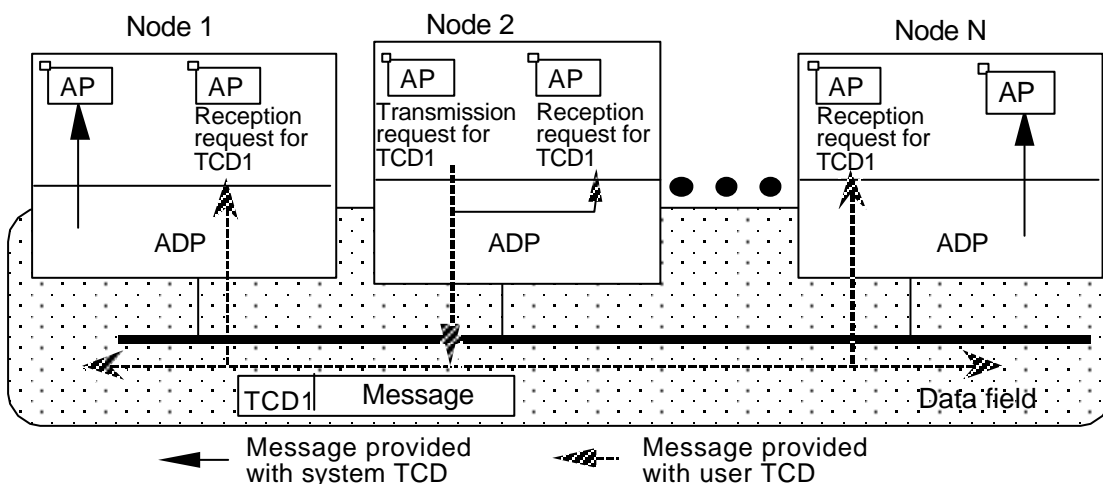


Figure 9 Transaction code

3.17 Implementor

An implementor refers to an enterprise or person that implements the ADP on a device.

3.18 User

A user refers to an end user who combines ADP-mounted devices to construct and use a system.

4 Protocol Functions

4.1 Class Base-1 (Multi-cast Communication)

4.1.1 Function

The transmitter sends a message associated with a TCD on a multi-cast transmission to a multi-cast group within a specified data field. The nodes within the specified multi-cast group automatically accept only the messages with required TCDs. Since this method does not specify any target address, extensible communication is available.

Before receiving the message on the multi-cast transmission, the nodes must have been added to the multi-cast group. If sending messages to all the nodes within one data field is required, all the nodes must have been added to the same multi-cast group.

NOTES (features of multi-cast communication):

- Provides communication platforms for autonomous decentralized system.
- Efficient method allowing simultaneous transmission of one and the same message to more than one node (one-to-multiple communication)
- Suitable for real-time data communication.
- Effective when transmitting periodically-generated-data to more than one node at once.

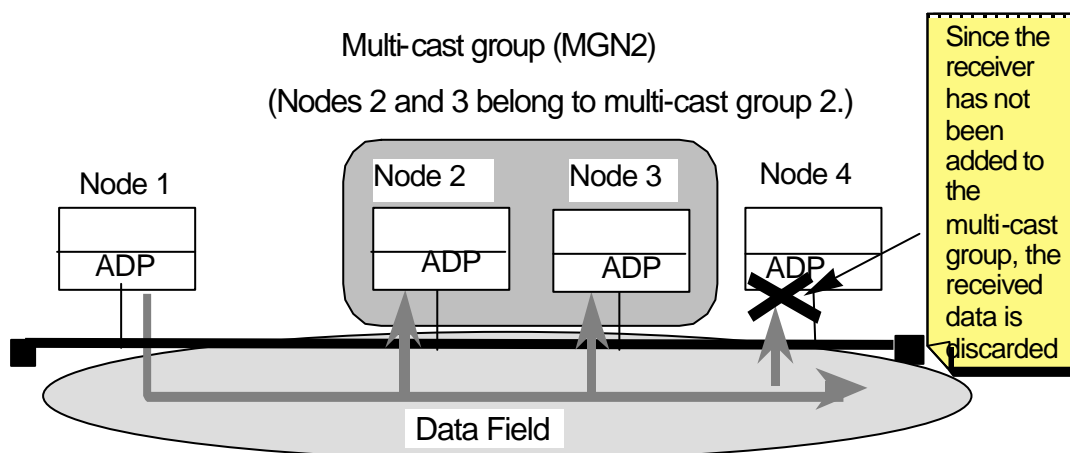


Figure 10 Multi-cast communication

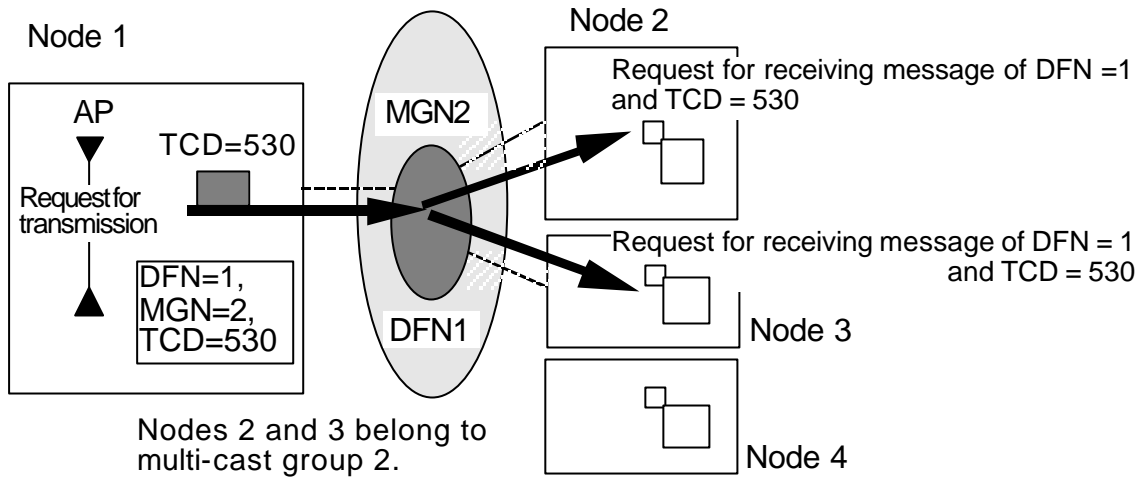


Figure 11 Sending/Receiving multi-cast communication

A multi-cast communication is the upper protocol of the UDP protocol. Data sent over a multi-cast transmission is broadcast to a LAN and received by nodes. If the current node has not been added to the target multi-cast group of the received message on the ADP level, the message is discarded, while if the node has been added to the group, ADP passes the message to AP.

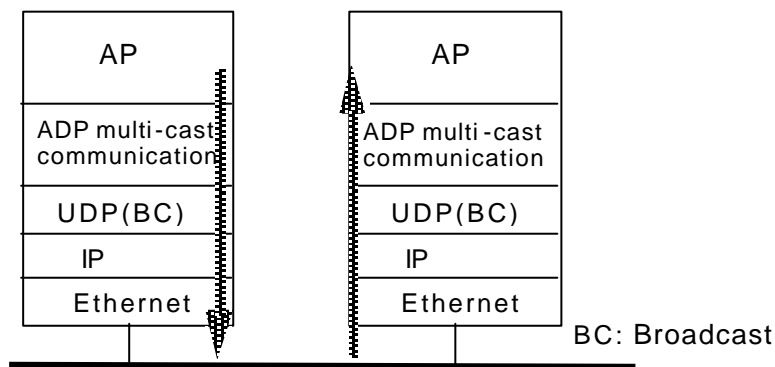


Figure 12 UDP-based communication

4.1.2 Managing Multi-cast Group

A multi-cast group is the group of the nodes defined within a data field. Each of the nodes can be added or removed from the multi-cast group as necessary.

4.1.2.1 Assigning MGN and UDP Port Number

MGNs can be assigned with numbers between 1 and 255. Multi-cast number 0 is reserved by the system for the activity report. A UDP port must be assigned to each message mode received by multi-cast groups

(see "4.5 Test Support").

The relationships between the message modes of MGNs and UDP port numbers must be consistent within a data field and managed by data field basis.

4.1.2.2 Adding and Removing to/from Multi-cast Group

4.1.2.2.1 Addition

A node is associated to a UDP port number identifying a multi-cast group when an event including starting the system occurs, and added to the multi-cast group.

Example: Adding procedure for a device using a socket

- (a) Generate a socket for multi-cast MGN_i with the Internet protocol and datagram type.
- (b) Bind the node to the socket with the wildcard address (IP address) and the UDP port number assigned to MGN_i.
- (c) Tune the buffer size of the receiving socket based on the received traffics.

4.1.2.2.2 Removing

This section describes how to release the connection between the node and UDP port number made when adding to the multi-cast group. Releasing the connection removes the node from the multi-cast group.

Example: Removing procedure for a device using a socket

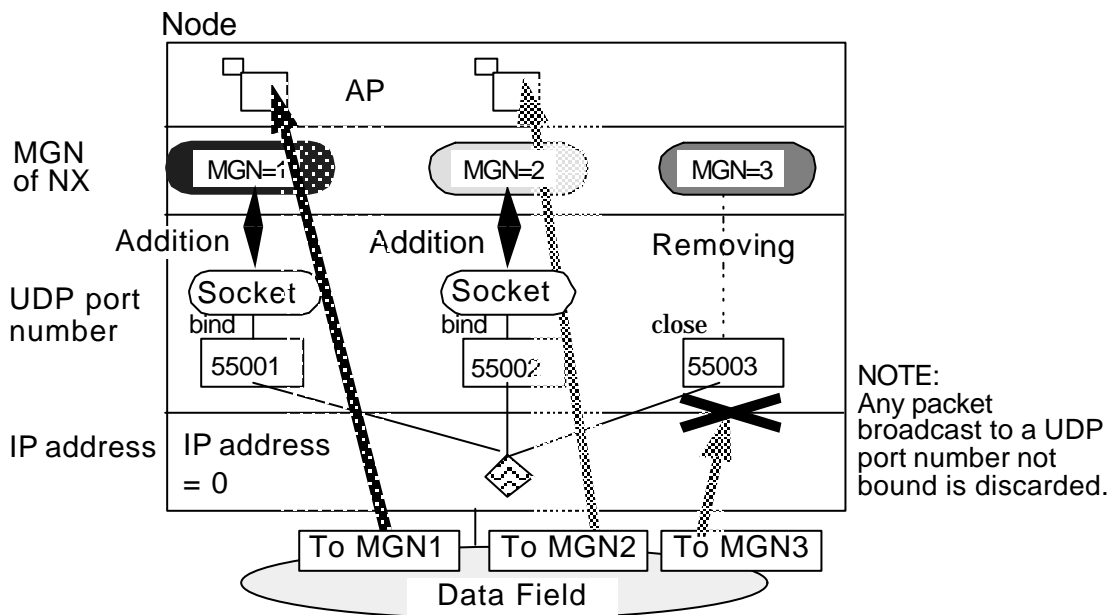


Figure 13 Adding and removing to/from multi-cast group

4.1.2.3 Creating Multi-cast Transmitting Port

This section describes how to assign a source UDP port number used for sending to a multi-cast group to a data field unit.

NOTE:

It is recommended to assign such source UDP port numbers that they allow one port to send more than one multi-cast group or allow different port numbers to be assigned to data fields. Also it is recommended to use one and the same source UDP port number for every node within one data field.

Example: Creating procedure for a device using a socket

- (a) Generate a socket for multi-cast sending with the Internet protocol and datagram type.
- (b) Bind the node to the socket with the wildcard address (IP address 0) and the source UDP port number.
- (c) Declare (setsockopt) that broadcasting is to be used for the socket.

4.2 Class Base-2 (Transmitting Alive Signal)

4.2.1 Function

A node transmits alive signals to the other nodes on the same data field to notify that the node is alive. This makes the nodes on the same data field can monitor the status to each other. All the nodes belonging to one data field must transmit the alive signals within the data field periodically as far as they can transmit the alive signals.

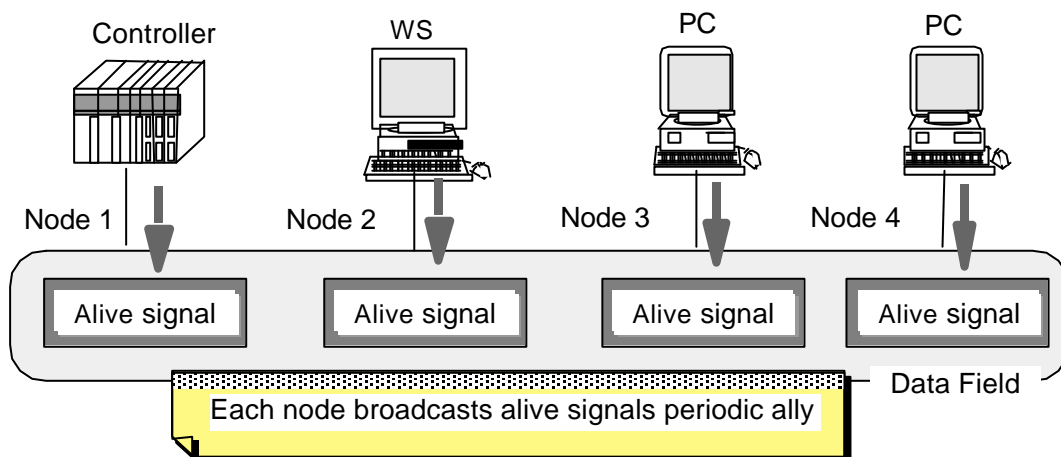


Figure 14 Transmitting Alive Signal

NOTE:

The alive signals are used to manage the node configuration. When one node on the same data field receives a transmitted alive signal, it can know that the source node is alive. This makes a node belonging to a data field can monitor the status of the other nodes (alive/dead, normal/abnormal) on the same data field to manage or control the whole system configuration.

Example:

Assuming that there is node 2, which backs up node 1 for its applications by monitoring the alive signals from node 1. If node 2 determines node 1 is dead, node 2 can undertake the applications on node 1, while if node 2 determines node 1 has been resumed again, node 2 can discontinue the applications undertaken from node 1.

4.2.2 Types of Transmitting Alive Signals

4.2.2.1 Extent of Target Data Field

Nodes transmit alive signal messages only within the data field associated to the network they are directly connected to. This means that alive/dead judgement by monitoring the alive signals is performed locally for each data field, therefore no alive signal message is transferred over routers to a remote data field. A node connected to more than one data field transmits alive signal messages for each data field.

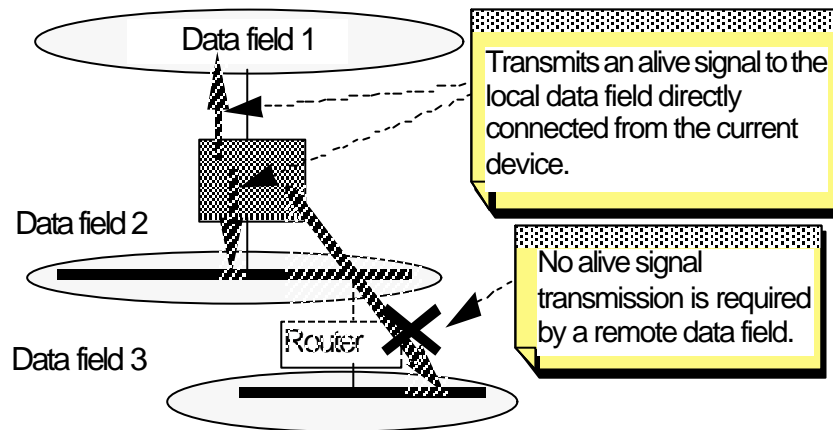


Figure 15 Target data field for alive signal transmission

4.2.2.2 Intervals for Transmitting Alive Signals

The intervals for transmitting alive signals must be set for each logical node. For nodes belonging to more than one node, the intervals must be set for each data field. After a device was turned to online, it transmits alive signals at the specified intervals.

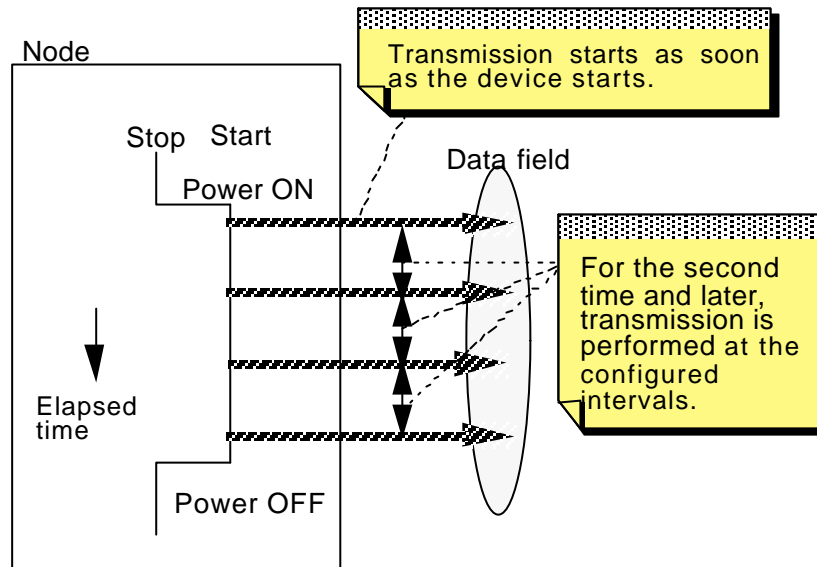


Figure 16 Intervals for transmitting alive signals

4.2.3 Alive/dead Judgement with Alive Signals

Received alive signals are used by a receiving node to determine whether a source node is alive. The receiving node determines whether the source node is alive based on the following conditions:

- If a node has received no alive signal from another node, the source node is determined to be dead.
- When a node receives an alive signal from another node, the source node is determined to be alive.
- If a node has received an alive signal from another node to determine the source node to be alive but it does not received a new alive signal for the duration of the alive signal delay time (al_tm_out) specified in the alive signal header for the received alive signal, the source node is determined to be dead. For details on the process, see section "D.5 Monitoring Node Alive" of the attached document.

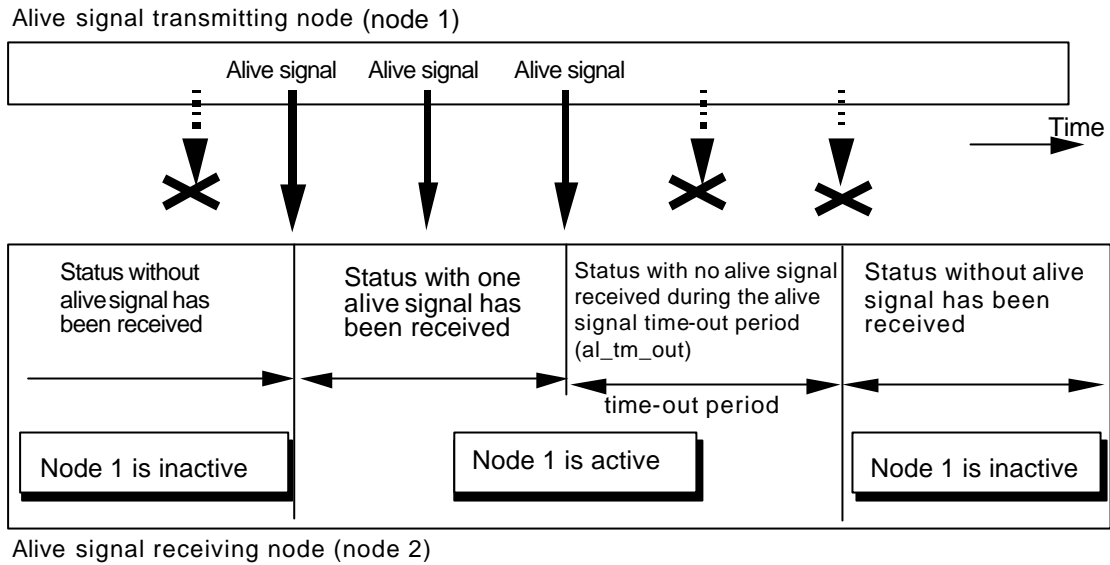


Figure 17 Node status monitoring sequence based on alive signal

4.2.4 Assigning UDP Port Numbers

Alive signal messages are sent on multi-cast to the local data field of the current device. Alive signals use multi-cast group 0. The group is reserved for broadcasting where all nodes receive the alive signals unconditionally, therefore, 0 is assigned to the target multi-cast group number.

For each device, a user must be able to assign any number to the target UDP port for transmission. For each data field, user may select any number for the UDP port used by alive signals.

NOTES:

- Nodes connected to the same data field must assign one and the same number to UDP ports for alive signals (see Figure 18).
- When the current node receives alive signal messages, a UDP reception port associated to the port number must be generated for the received messages. For the receiving UDP port, assign the UDP port number specified in the corresponding data field (see Figure 19).

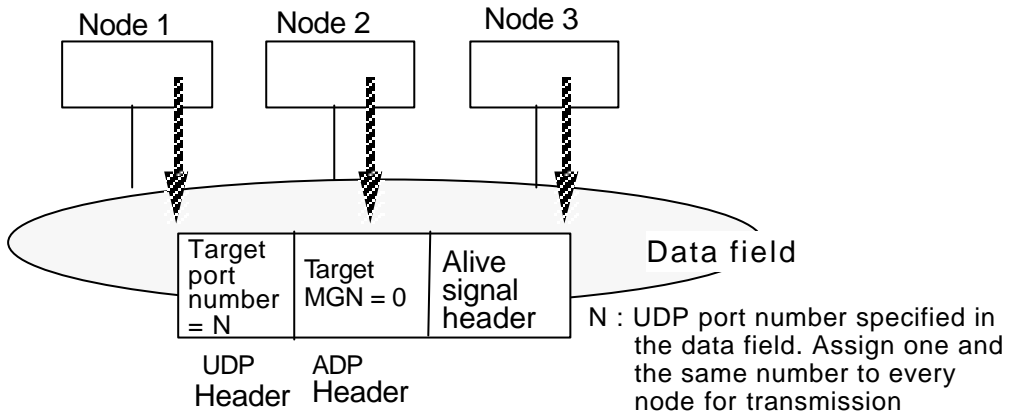


Figure 18 Target UDP port number for alive signal

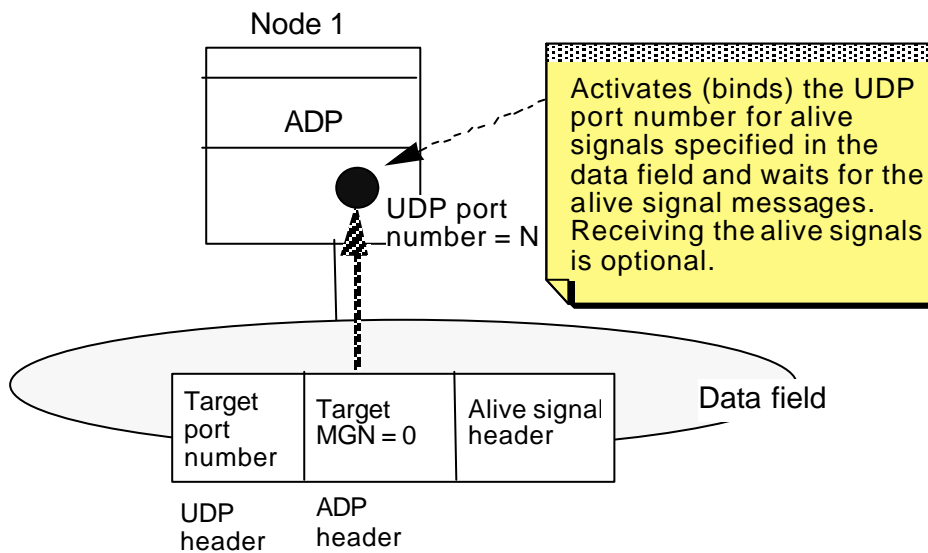


Figure 19 Receiving UDP port number for alive signal

4.3 Class Opt-2-a (Transmitting Fault Information)

4.3.1 Function

This can transmit fault information as the extended information of alive signals. When a fault condition occurred on a node within a data field, this function notifies the other nodes in the same data field of the information on the fault.

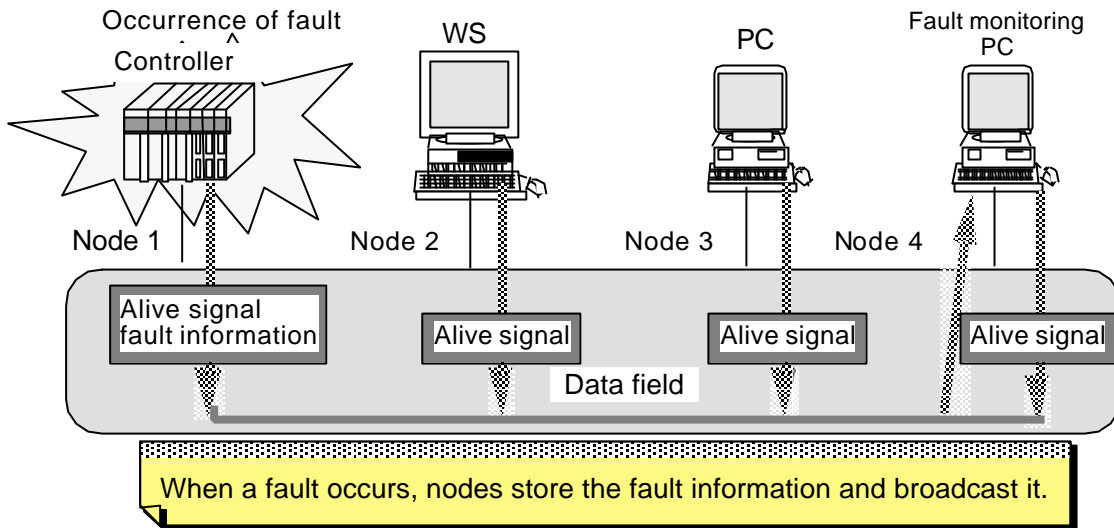


Figure 20 Transmitting fault information

4.3.2 Fault Information

When a node receives fault information it converts it into the following items:

- Information on activity of modules in node

This indicates the module status when transmitting.

NOTE:

The above module may be either of a software or hardware module. A module on a node is assumed to be a source of transmitted activity information (i.e., a target of activity monitoring). Software modules generally include processes and tasks.

- Fault number (error number)

A fault number identifies the associated fault occurred within a node. It only reports on the fault after the last transmission. If the fault has not been recovered and the system stays in the fault status, the fault number continues to be submitted. Transmitting a fault number(s) enables a monitoring application to indicate the contents of the fault (see section F.2 of the attached document). This also allows the node receiving fault information to identify the contents of the fault occurred on another node.

- Optional information unique to device

This indicates the fault information unique to a device in a format unique to the device.

Example:

Outputting the contents of the status register in a controller enables the monitoring application described later to display the contents (see section F.2 of the attached document).

4.4 Class-Opt-3 (Peer-to-Peer Communication)

4.4.1 Function

The peer-to-peer communication is based on TCP. Before communication, one peer-to-peer communication path (TCP connection) must be established. When AP requests for transmission, a message is sent to the specified target node. Before the TCP connection can be used, an IP address and connection port number must be configured appropriately for each target node.

This communication method is useful when data must be sent securely to the target node on the one-to-one basis or bulk data transmission is required over a peer-to-peer communication path.

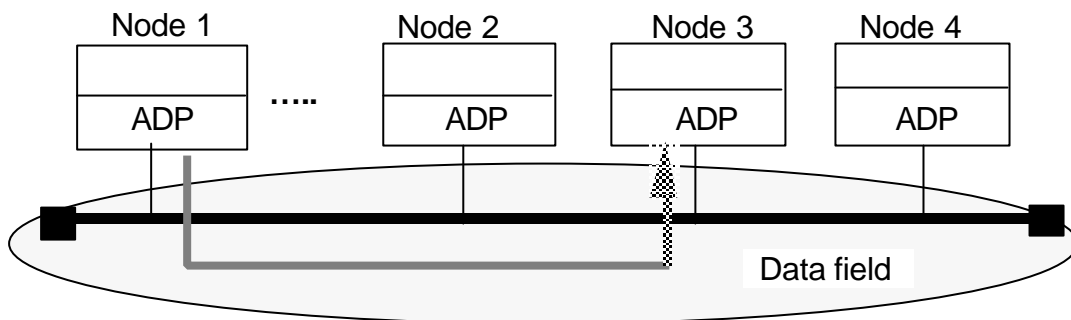


Figure 21 Peer-to-Peer communication

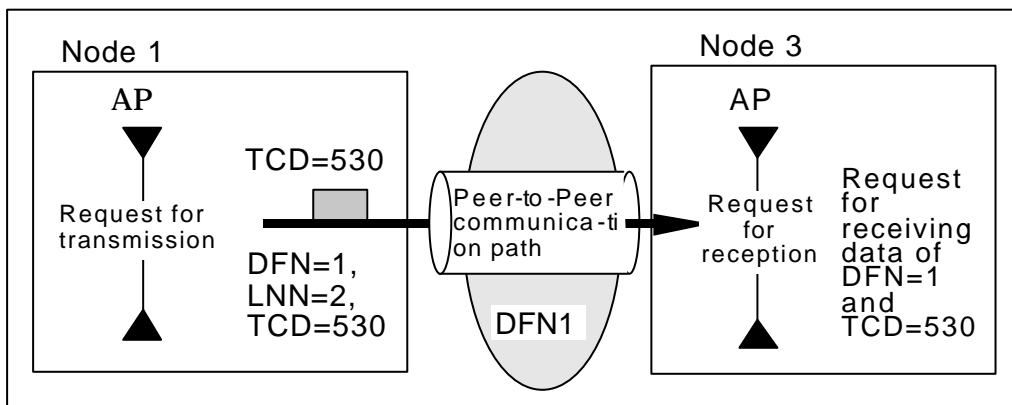


Figure 22 Peer-to-Peer communication path and transmission/reception

Peer-to-Peer communication positions upper of TCP. TCP between nodes transfers a transmitted peer-to-peer communication message through a TCP connection.

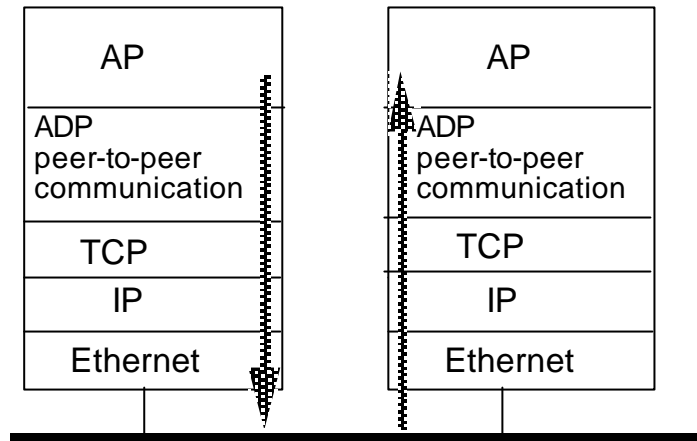


Figure 23 TCP-based communication

4.4.2 Managing TCP Connection for Peer-to-Peer Communication

4.4.2.1 Connection and Assigning TCP Port Number

For a peer-to-peer communication, a TCP port unique within a node must be assigned. When a peer-to-peer connection is established, a TCP port number defined between the two nodes is used.

Example:

Assuming that node 1 wants to connect to nodes 2 and 3 respectively. For each of the connections, an IP address to the target node number and a target TCP port number are necessary.

As the connection information between nodes 1 and 2, IP addresses IP1 and IP2 and TCP port numbers 10000 and 20000 are used. For nodes 1 and 2 connections, the four pieces of information must be defined respectively.

Before a connection can be established, the requester and receiver of the connection request must be specified. The requester may fail in establishing a connection due to any cause such as an inactive target node, however the requester should be able to retry requesting periodically. As a connection established, peer-to-peer communication becomes available.

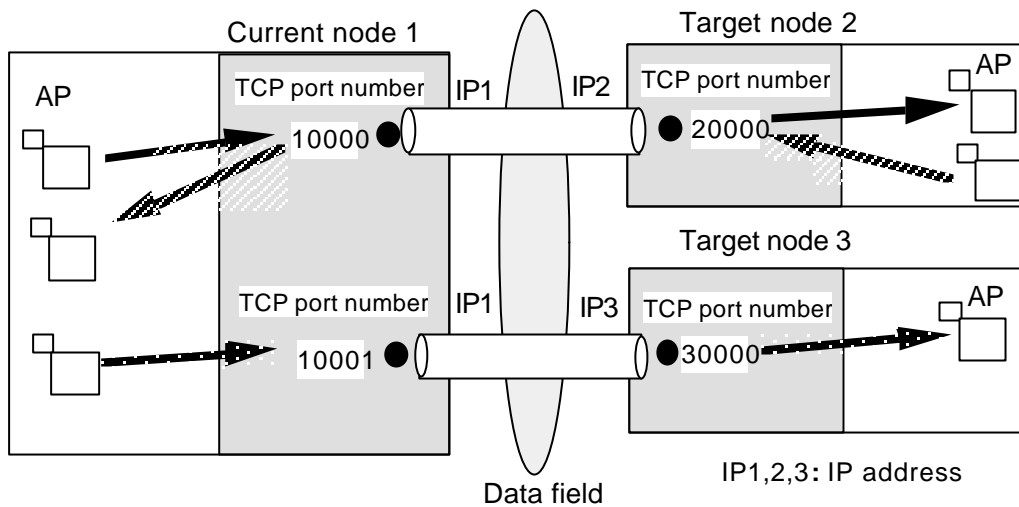


Figure 24 TCP connection and assigning TCP port numbers

4.4.2.2 Peer-to-Peer Communication Status

ADP manages the connection status for a peer-to-peer communication based on the following three statuses:

- Halt status

In this status, the target communication node has been known by the system, and the system management table for the related nodes has been initialized. This status allows no message to be exchanged but a connection with another node to be established. The Halt status indicates the TCP status, CLOSED.

- Open status

This indicates that the connection is under transition, on the way of establishing a connection. As a connection is established, this status automatically replaced with the link status. This status allows no message to be exchanged. On the way of establishing a connection means a status where establishing a connection with the target node has been requested, and a response from the target is waited (the TCP status of SYN_SENT) or where a request for establishing a connection with the target node is being waited (TCP status of LISTEN or SYN_RECEIVED).

As the connection with the target is established (TCP status of ESTABLISHED), the link status is entered. If establishing a connection is failed, the open status is kept. If moving to the link status is instructed with keeping the open status established, the halt status must be entered temporarily before the link status can be entered.

- Link status

This indicates that the connection with the target node has been established and transmitting/receiving message is available (TCP status of ESTABLISHED). If the system instructed to move to the halt status,

the both nodes automatically move to the halt status. Other TCP products generally issue FINs and use the three-way handshake sequence to release a connection, while this system uses a RESET packet to cut a connection instantaneously. If a connection cut (receiving a RESET) or communication fault is detected under the link status, the system automatically enters the halt status.

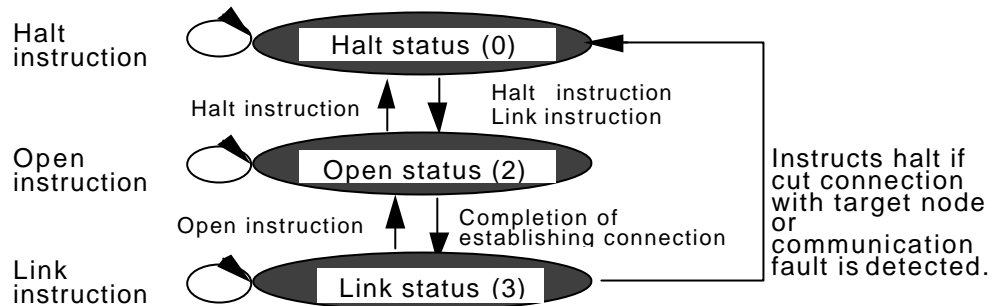


Figure 25 Status transition for TCP connection

4.5 Test Support

4.5.1 Function

ADP allows adding modes to a message to indicate whether it is online or under testing. Since a node can control transmitting/receiving messages selectively based on the discriminant mode (online/testing), the node can eliminate any interrupting test message when it is on the way of testing. When a node is performing a simulation test, it can receive either of the online or test message.

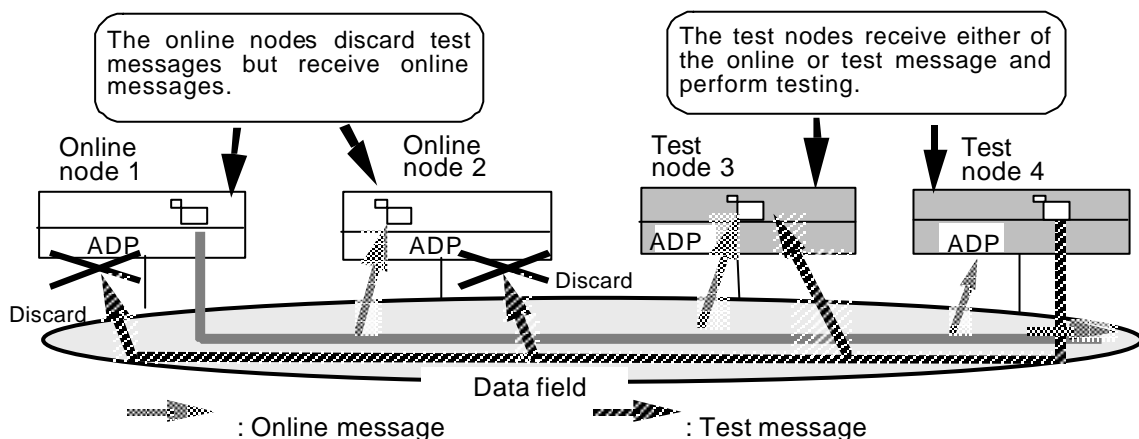


Figure 26 Test support

4.5.2 Message Modes

There are two types of message modes:

- (1) Online
- (2) Test

4.5.3 Node modes

There are two types of node modes:

- (1) Online
- (2) Test

4.5.4 Message Transmitting/Receiving Control

The following table shows the relations between the message and node modes that can be used to control message transmission/reception.

Table 2 Relation between message transmission/reception

Node mode \ Message mode	Online		Test	
	Transmission	Reception	Transmission	Reception
Online	○	○	✕	○
Test	✕	Discard	○	○

○ : possible

✕: impossible

4.5.6 Assigning Multi-cast Receiving Ports

To receive only online mode messages at an online node, use one UDP port per multi-cast group.

To receive either of online or test mode messages at a test node, use one UDP port for online messages and another for test messages.

NOTE:

When associating different UDP ports for online and test mode messages within one multi-cast group, consistency must be retained across the nodes within one data field.

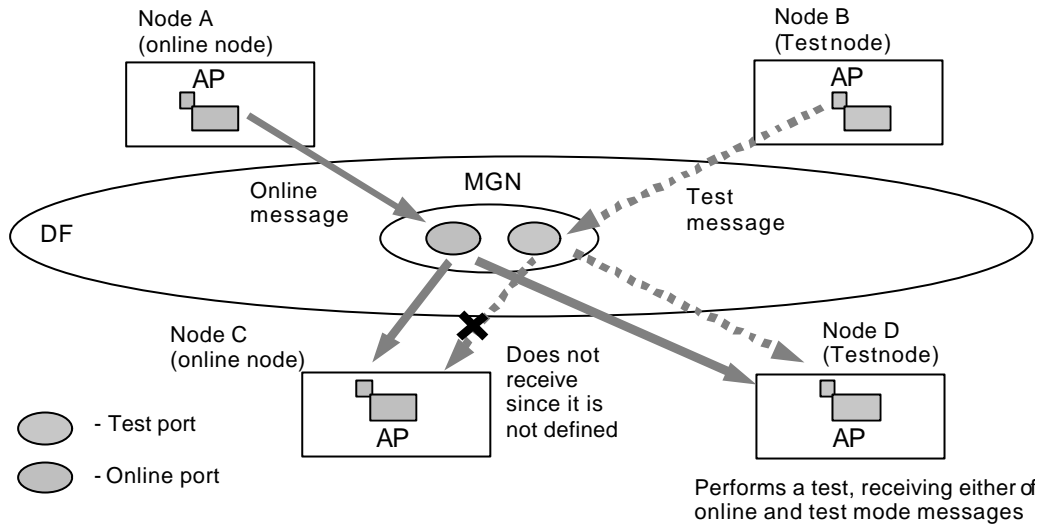


Figure 27 MGN and reception port

This avoids test mode messages from disturbing online nodes even if online and test nodes can reside on the same data field, since different ports are used for online and test mode messages. Especially when using devices that do not receive messages through undefined ports on a hardware level, it is effective to relieve the CPU load.

Example:

Figure 28 shows the relations between multi-cast group numbers, 1 to 255, and UDP numbers (online and test modes).

Multi-cast group number	1	2	3	255	← Target MGN
UDP port number (online)	55001	55002	55003	55255	} UDP protocol level targets Port numbers
UDP port number (test)	57001	57002	57003	57255	

Figure 28 Assigning MGN and UDP port numbers

4.5.6 Operations of Test Support

(1) Transmitter Side

The transmitting node passes the mode of a message to be transmitted into the message mode (MODE) in the ADP header and sends the message.

(2) Receiver Side

If the current node is in the online mode, the reception node checks the message mode (MODE) in the ADP header and discards any test message but receives online messages. Since discarding a test message is not assumed to be a problem, it is not necessary to issue a fault information for discarding. If the current node is in the test mode, either of test or online message is accepted.

4.6 Controlling Message Priorities

4.6.1 Function

This prioritizes messages to schedule transmission and reception of them. This avoids a message with a lower priority from interrupting the transmission or reception of a message with a higher priority.

4.6.2 Message Priority Levels

There are seven message priority levels as shown in table 3. To nodes with no message prioritization implemented, node message priority 0 is assigned.

Table 3 Priority level assignment

Level	Priority	Use	Operator
0	None	Out of priority control	Users
1	High	Message with highest priority for users	
2	↕	...	
...			
7	Low	Message with lowest priority for users	

4.6.3 Prioritizing Messages

4.6.3.1 Transmitter Side

- The transmitting node assigns one of priority levels 1 to 7 depending on the priority of the transmitted message, passes the specified message priority level to the message priority level (PRI) in the ADP header of the message's PDU and sends the message.
- If the transmitting node is not implemented with the message prioritization, it passes 0 to the message priority level (PRI) in the ADP header and sends the message.

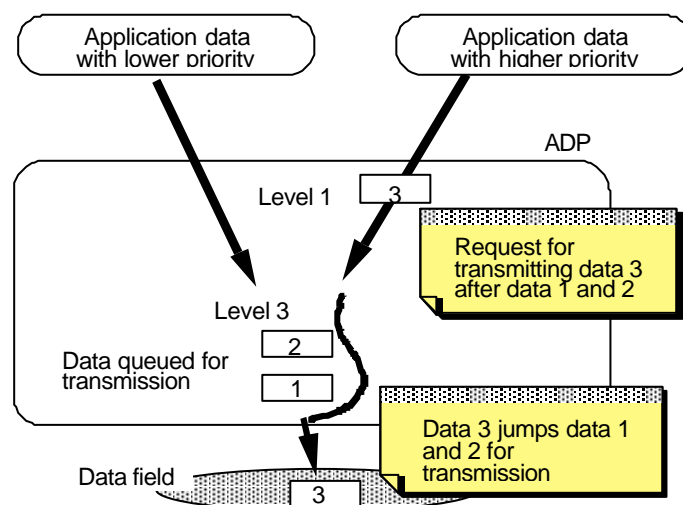


Figure 29 Message prioritizing at transmitter

4.6.3.2 Receiver Side

- The receiving node checks the message priority level (PRI) referring to the ADP header of the received message. If the level is within 1 to 7, the node receives the message based on the specified priority level and pass it to AP.
- If the node receives a message with priority level 0, it converts the priority level based on the priority level defined previously for priority level 0 message and processes it. The lowest priority level is generally used for processing a priority level 0 message.

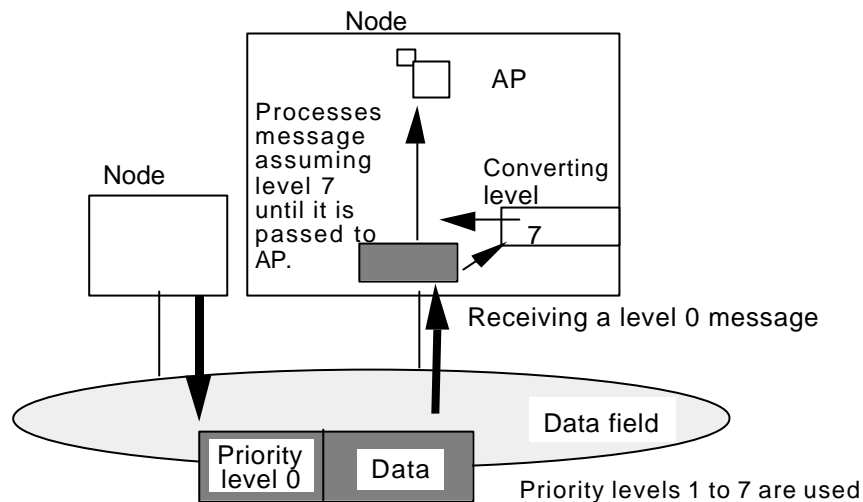


Figure 30 Message priority control at receiver Priority level conversion for priority level 0

4.7 Sequential Numbering for Message Management

4.7.1 Function

The receiver numbers the message transmission sequentially and makes versioning for the sequential numbering to eliminate missing messages.

4.7.2 Message Transmission Sequential Number

Messages are numbered sequentially to detect a missing message.

4.7.3 Versioning of Message Transmission Sequential Number

This is used to detect resetting the sequential numbering for message transmission. It is generated whenever the sequential numbering for message transmission is initialized.

NOTE:

If the transmitting node is shut down and the sequential numbering is reset, the receiving node may misunderstand that the sequential numbering was missed. To avoid this problem, versioning is performed on

the sequential numbering of message transmission. It is recommended to use time stamp for versioning. Figure 31 shows an example of using the time stamp for versioning and how it works.

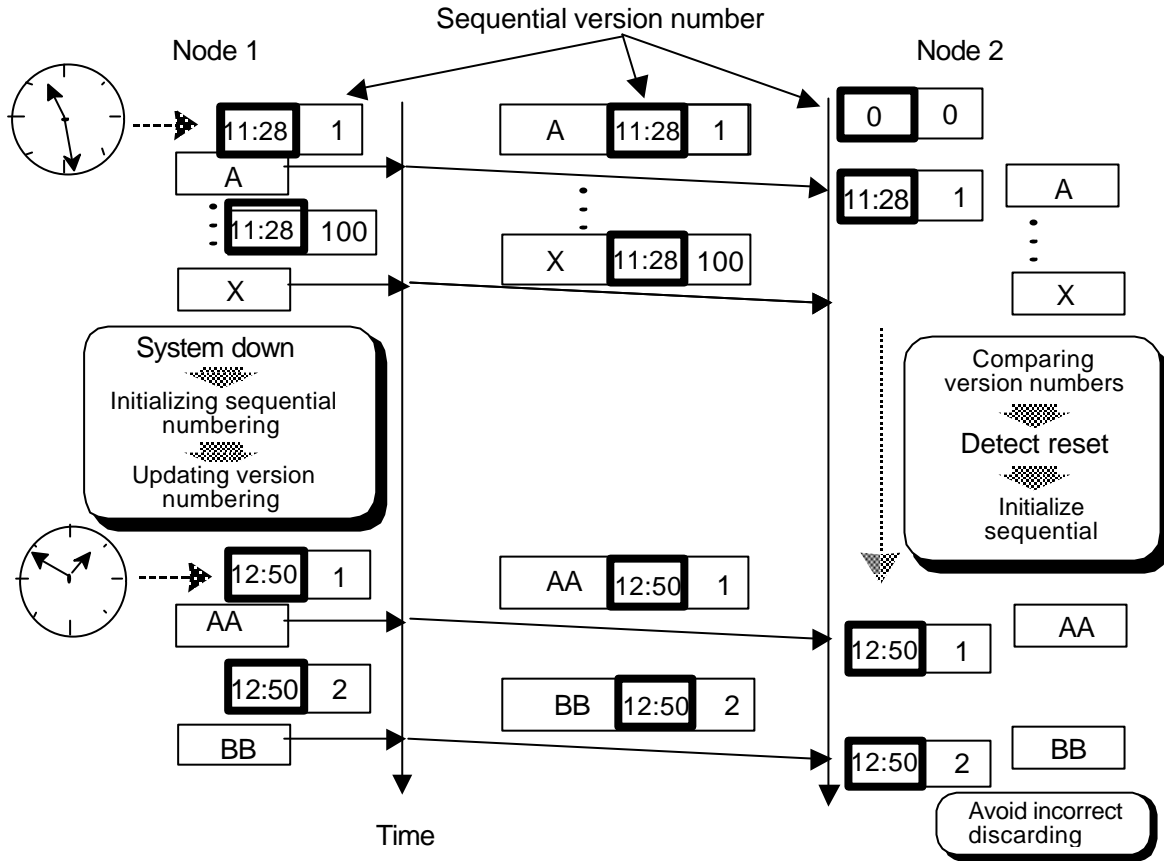


Figure 31 Versioning of sequential number

4.7.4 Managing Message Sequential Numbers

4.7.4.1 Transmitter Side

The transmitting node maintains the message transmitting sequential numbers and versioning of the sequential numbers. When transmitting a message, it passes the message transmission sequential number (SEQ) of the message's PDU and the versioning of the message sequential numbers (V_SEQ), and sends the message.

4.7.4.2 Receiver Side

The receiving node maintains the most recently received sequential number and the most recently received versioning of the received sequential number for each message source node. It checks the message transmission sequential number (SEQ) and the versioning of the message sequential numbers (V_SEQ) in the ADP header of the received message's PDU. If the versioning of the sequential numbers of the received message differs from the most recent versioning of the sequential number, the receiving node assumes resetting has occurred on sequential numbering, assumes the most recently received sequential number as the sequential number of the received message.

If the versioning of sequential number matches with the most recent versioning of the sequential number but the sequential number of the received message has been missed, it is assumed to be omission of sequential numbering.

For details on the sequential numbers and versioning of sequential numbers for the multi-cast and peer-to-peer communications, see section D.3 of the attached document.

4.8 Dividing and Combining Message

4.8.1 Function

If the total of the TCP/UDP, IP protocol, ADP header and message body sizes is larger than MTU, the message transmitter divides the message into more than one PDU before transmission. When the message receiver receives PDUs, it combine them into the original one message.

4.8.2 Information for Dividing and Combining Message

The following are the information used when performing dividing and combining on a message.

CBN: Current fragmented PDU number (PDU number starts from 1)

TBN: Total number of fragmented PDUs

Bsize: PDU size

ML: Message length + 64 (in byte)

SEQ: Sequence number of message

The transmitting node passes the above information to the ADP header of the message's PDU. The receiving node checks the ADP header of the received message's PDU to combine the fragmented message components.

For details on performing dividing and combining on a message for the peer-to-peer communication, see section D.4 of the attached document.

4.9 Associative Array Protocol

4.9.1 Message structure

The information describing data items comprising a message and values in the data items together form application data.

The information describing data items includes names and types, and the protocol draft uses ASN.1 (IEC8824/8825) as encoding rules and the data types regulated in IEC61131-3 as the basic data types.

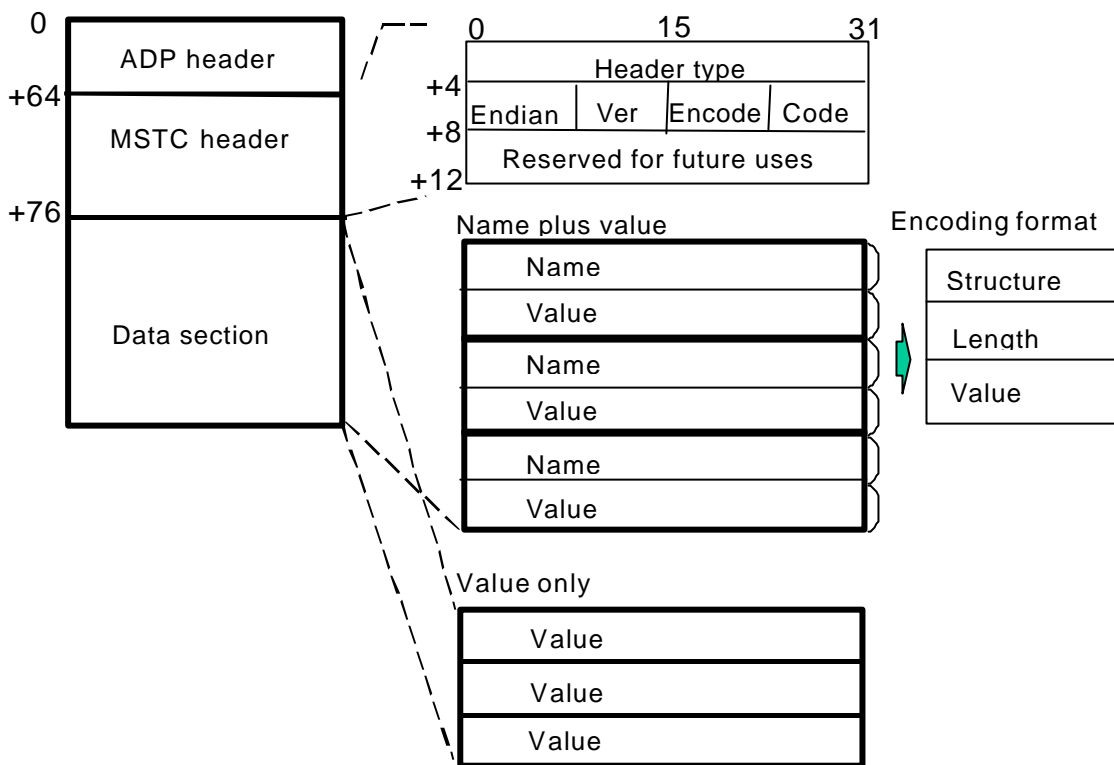


Figure 32 Message structure

MSTC header: Indicates the structure of data section. Specify in the big endian.

Table 4 Header format

Item name	Description
Header type	Header identifier in ASCII. Fixed to "MSTC."
Endian	Endian type of the data value in the data section. 1 = Little, 2 = Big, 3 = None
Ver	Version of the protocol. Fixed to 1.
Encode	Indicates the use of data section. 1 = Data value only (general type) 2 = Data value and item information (integrated type with item description, and is the draft type)
Code	Indicates the code scheme of the string used in the data section

4.9.2 Structure representation

4.9.2.1 Basic data type (for details, see IEC61161-3.)

Table 5 Basic data type

Data type name	Data type code (in hex.)	Description
BOOL	C1	Logical value(TRUE/FALSE)
SINT	C2	Signed 8bit integer
INT	C3	Signed 16bit integer
DINT	C4	Signed 32bit integer
LINT	C5	Signed 64bit integer
USINT	C6	Unsigned 8-bit integer
UINT	C7	Unsigned 16-bit integer
UDINT	C8	Unsigned 32-bit integer
ULINT	C9	Unsigned 64-bit integer
REAL	CA	32-bit floating point
LREAL	CB	64-bit floating point
STIME	CC	Information on synchronizing time
DATE	CD	Date
TIME_OF_DAY	CE	Time
DATE_AND_TIME	CF	Date and time
STRING	D0	8-bit string
BYTE	D1	8-bit column
WORD	D2	16-bit column
DWORD	D3	32-bit column
LWORD	D4	64-bit column
STRING2*	D5	2-byte string
FTIME*	D6	Duration [High resolution]
LTIME*	D7	Duration [Long precision]
ITIME	D8	Duration [Short precision (same as the INT type)]
STRINGN*	D9	N-byte string
SHORT_STRING*	DA	8 bit (1-byte flag)

* Any item marked with an asterisk (*) is an extension not comprising with IEC61131-3.

4.9.2.2 Structure and array representations

For structures and arrays, two types of structure/type shall be provided: one with a name and value for

each component and the other with only a value for each component.

Table 6 Structure and array representations

Data type name	Data type code (in hex.)	Description
ANY_STRUCT	E0	Structure or array with each component having a name (including a type and length) and value (including a type and length) pair
SEQ_STRUCT	E1	Structure or array with each component having only a value (including a type and length).

4.9.3 Encoding example

4.9.3.1 Data items of basic data type (with a name and value)

Example) UINT item=5;

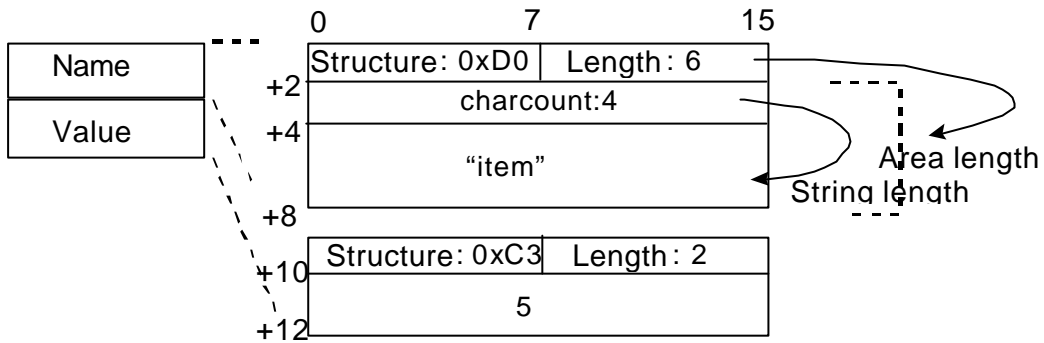


Figure 33 Encoding example 1

4.9.3.2 Data with time attribute (with a name, value and time information)

Example) ANY_MAGNITUDE ::= {
 ANY_NUM
 TIME_OF_DAY
 }

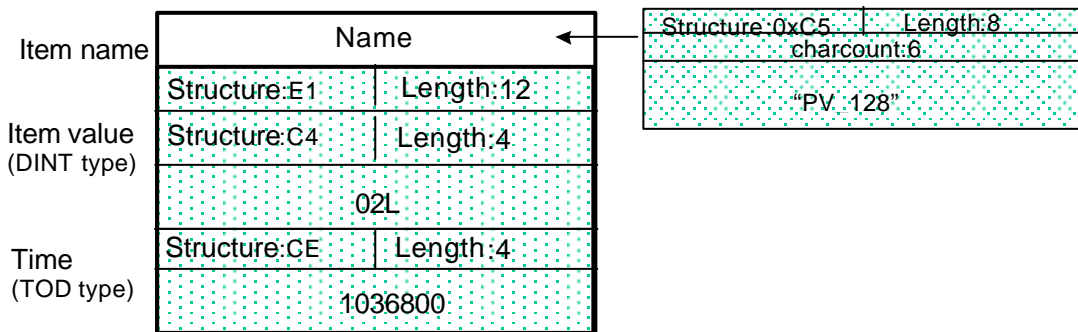


Figure 34 Encoding example 2

4.9.3.3 String array

Example) STRING str2[2]

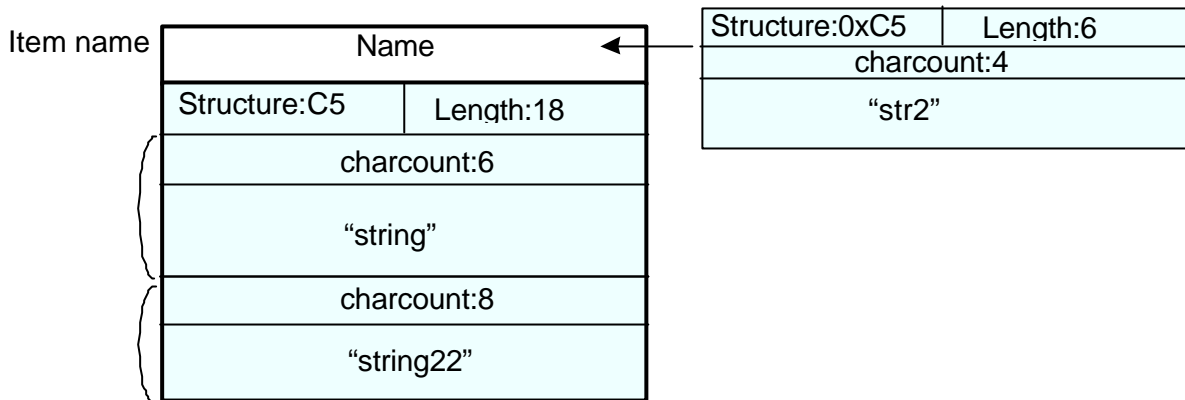


Figure 35 Encoding example 3

4.9.3.4 Data items of array and structure

Example: struct { DINT item01; DINT item02[2]; } struct1 = { 01L, { 02L, 03L } };

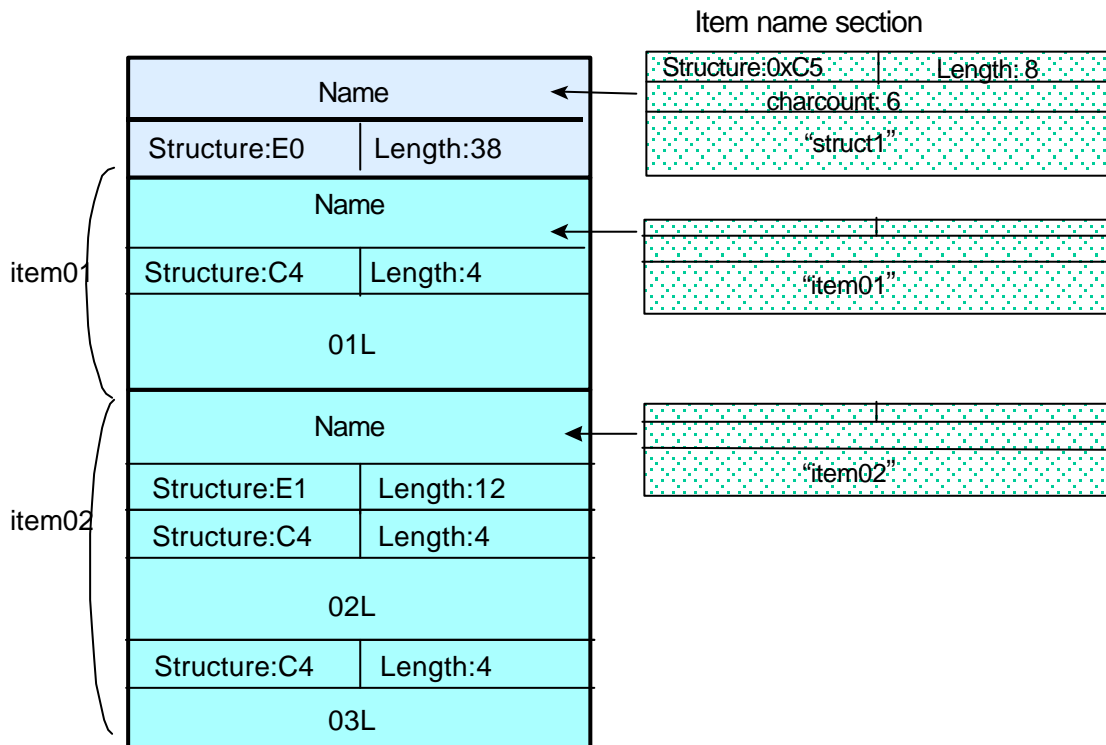


Figure 36 Encoding example 4

5 PDU Structure and PDU Encoding

5.1 PDU Structure

Figure 37 shows the structure of a Protocol Data Unit (PDU). A PDU consists of a fixed length ADP header section and a variable length data section. Table 7 shows the information contained in the ADP header section.

The relative address of the message and header indicates the data positions when flowing across a network. The bit positions are indicated by 0- to 31-bit in order when flowing into a network. Hereafter in this document, these representations are used for the message formats and header representations. The autonomous decentralized protocol uses the big endian for the bite order.

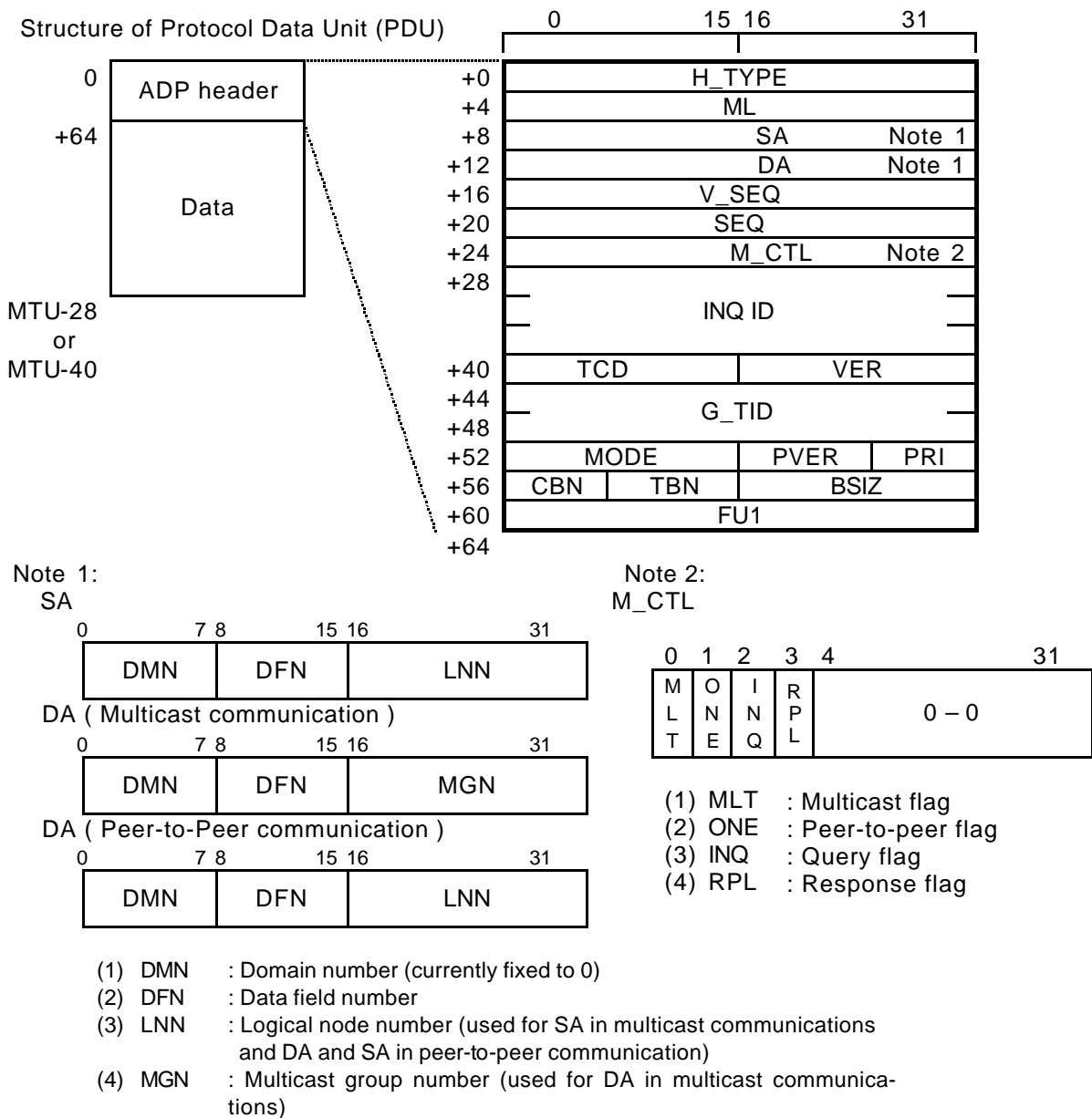


Figure 37 PDU structure and ADP header

Table 7 Information contained in ADP header

No	Identifier	Size	Description
1	H_TYPE	4	Header type. Specify "NUXM" in ASCII to distinguish the message from other protocol messages (fixed).
2	ML	4	Message length plus 64 bytes.
3	SA	4	Source message address. See NOTE 1 in Figure 37
4	DA	4	Target message address. See NOTE 1 in Figure 37.
5	V_SEQ	4	Message transmission version. This normally specifies the time when the message transmission number is initialized.
6	SEQ	4	Message transmission number. Valid between 0x00000001 and 0x7FFFFFFF, and used cyclically.
7	M_CTL	4	Message transmission control information. See NOTE 2 in Figure 37.
8	INQ_ID	12	Query response identifier (Fixed to 0 for future uses)
9	TCD	2	Transaction code
10	VER	2	Program version for updating the program (Fixed to 0 for future uses)
11	GTID	8	Transaction identifier (Fixed to 0 for future uses)
12	MODE	2	Message mode (0 = Online mode, 1 = Test mode)
13	PVER	1	Protocol version (Fixed to 1)
14	PRI	1	Message priority level (1 is assigned to the top priority between 1 and 7. Fixed to 0, if no priority control is implemented.)
15	CBN	1	Current fragmented PDU number (more than 1)
16	TBN	1	Total number of fragmented PDUs (more than 1)
17	BSIZE	2	PDU size (including header size)
18	FUI	4	Fixed to 0 for future uses

5.2 Class Base-1 PDU (for Multicast Communication)

Figure 38 shows the format of the Multicast Communication PDU and setting values for its ADP header. The data section contains communication data.

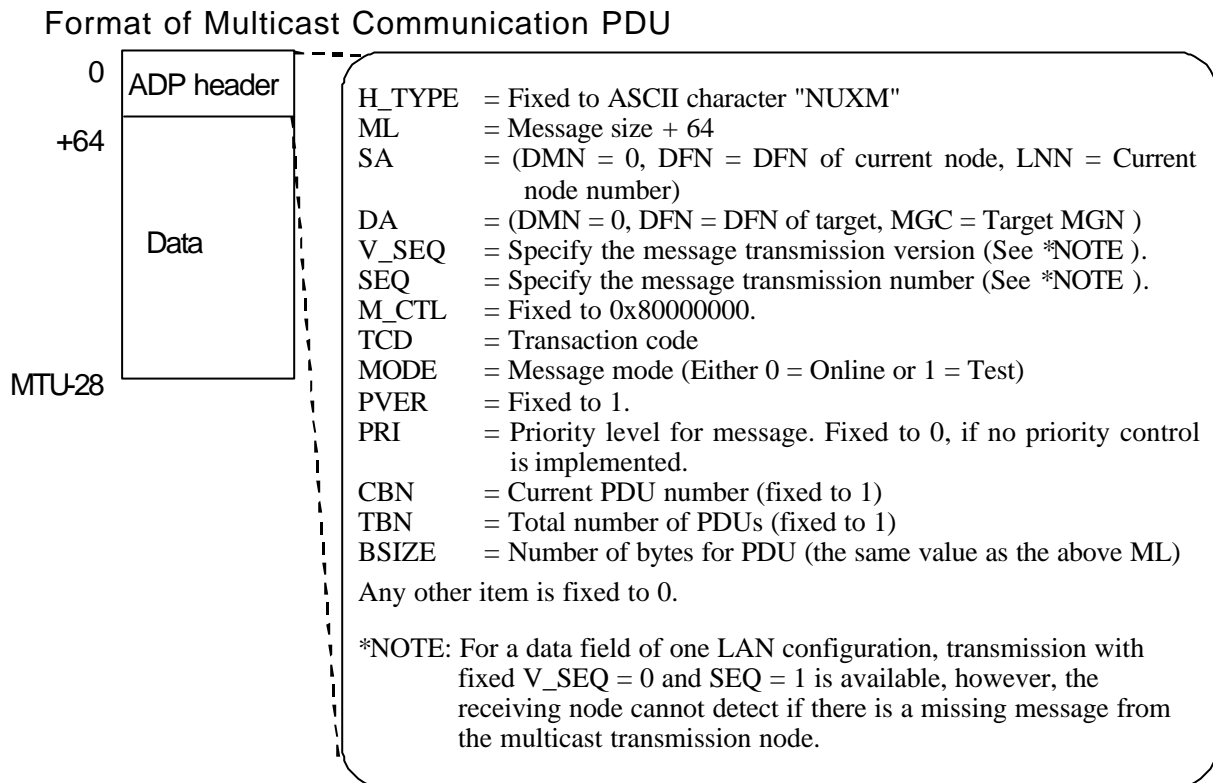


Figure 38 Format and ADP header setting values of Multicast Communication PDU

5.3 Class Base-2 PDU (for Alive Signals)

Figure 39 shows the format of the alive Signal PDU. The PDU's header section consists of the fixed-length alive signal header and variable-length fault information section. The alive signal header contains alive signal information.

Figure 39 also shows the setting values for the alive signal PDU and the structure of the alive signal header.

Table 8 shows the alive signal header information.

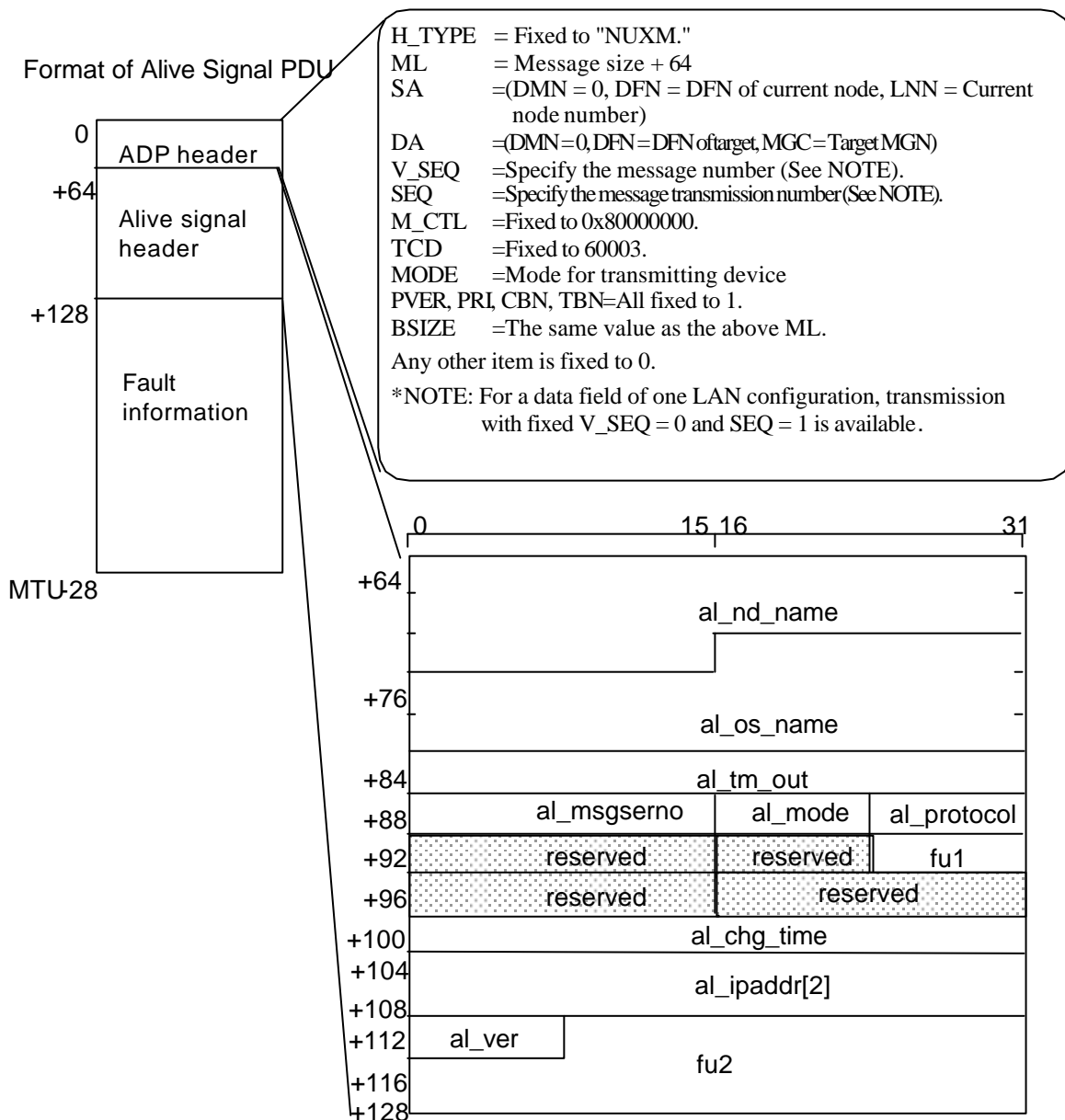


Figure 39 Alive signal message format and alive signal header

Table 8 Alive signal header

No.	Identification	Size	Description
1	al_nd_name	10	Node name (An ASCII string, up to nine characters ended with NULL)
2	al_os_name	10	Name of vender device (An ASCII string, up to nine characters ended with NULL). The following is the recommended naming format: "BN_MN" where BN: A code of vender name MN: A device name defined by a vender or an operating system name (up to six characters). For the vendor codes list, see Appendix B.
3	al_tm_out	4	The timeout delay for alive signals. This interval (in seconds) starts when the last alive signal is sent by a node and ends when the node is determined to be dead.
4	al_msgserno	2	The message number for a alive report. This parameter is only available for the duplex LAN control. Valid between 0x00000001 and 0x7FFFFFFF, and used cyclically. For a device not supporting the duplex LAN control, specify 0.
5	al_mode	1	Alive report mode. For the autonomous decentralized protocol in the specifications, alive signal transmission with "1" is mandatory. 1: Normal (Raw state) Normally this value is output. 2. Shutdown notice The notice is transmitted in this mode immediately before terminating the transmission of alive signals, to inform other devices that the coming alive signal shutdown will be caused by stopping a device. Transmitting in this mode is optional. 3. Maintenance notice The notice is transmitted in this mode immediately before terminating the transmission of alive signals, to inform other devices that the coming alive signal shutdown will be caused by a device maintenance. Transmitting in this mode is optional.
6	al_protocol	1	The type of the data section use following the alive signal header. In the autonomous decentralized protocol in these specifications, "4" is used. 1 to 3: Reserved (for other existing protocols). 4: Autonomous decentralized protocol in the specifications 5 and later: Reserved (for future uses)
7	fu1	1	Fixed to 0 for future uses.
8	al_chg_time	4	The time when a node status changed. It is recommended to set the time in S:M:H, D-M-Y of GMT (as the elapsed time since 0:0:0, 1970). If setting in GMT is unavailable, specify 0. A node status change is assumed as follows, and the values are used to indicate their associated node statuses: (1) al_mode = 1 starts to transmit an alive signal (indicating when the status shifts from the stop to alive statuses). (2) al_mode = 2 transmits an alive signal (indicating when a shutdown (device stop) request is received). (3) al_mode = 3 transmits an alive signal (indicating when a shutdown request for a maintenance is received).
9	al_ipaddr[0] al_ipaddr[1]	4	Specify the IP address of a LAN for al_ipaddr[0]. Specify 0 for al_ipaddr[1].
10	al_ver	1	Message version for alive reports (currently fixed to 1).
11	fu2	15	Fixed to 0 for future uses.
12	reserved	7	Specify 0.

5.4 Class Opt-2-a (fault information)

The fault information section of the Alive Signal PDU contains fault information (see section 5.3).

Figure 40 shows the structure of the fault information section and Table 9 shows the contents of the fault information.

Alive signal message <Fault information format>

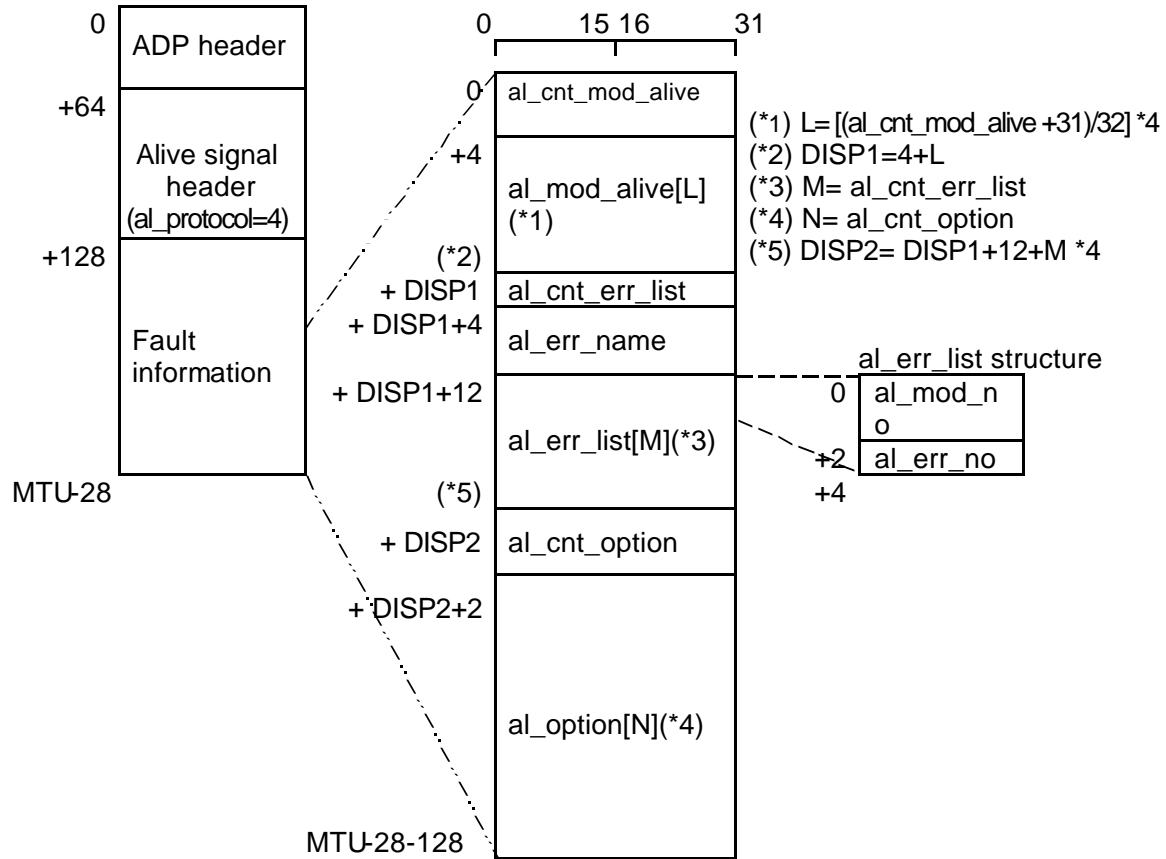


Figure 40 Format of fault information

Table 9 Contents of fault information

No	dentification	Size	Description
1	al_cnt_mod_alive	4	Number of alive report modules (*4)
2	al_mod_alive[L]	L (*1)	<p>Alive information on modules (*4)</p> <p>(1) If the module number (*5) X (starting from 1) is "alive," the position at X-1-[(X-1)/8]*8 bit of al_mod_alive [(X-1)/8] is cleared to 0.</p> <p>(2) If the module number (*5) X (starting from 1) is "dead," the position at X-1-[(X-1)/8]*8 bit of al_mod_alive [(X-1)/8] is set to 1.</p>
3	al_cnt_err_list	4	The number of reported errors. Up to [(MTU-28-128-4-4-8)/4]. (*2)
4	al_err_name	8	An error name for each error numbering scheme. The name is used by a monitoring tool as a prefix to identify an error message file name when it retrieves an error message associated to an error number.
5	al_err_list[M]	M*4 (*2)	An error information list
6	al_mod_no	2	A module number (*5) (starting from 1)
7	al_err_no	2	An error number. A unique number identified and embodied by al_err_name
8	al_cnt_option	4	The number of optional information bytes
9	al_option[N]	N (*3)	Optional information (Freely available area without a defined format)

(*1) $L = [(al_cnt_mod_alive + 31) / 32] * 4$

(*2) $M = al_cnt_err_list$

(*3) $N = al_cnt_option$

(*4) Module: A unit for which you want to identify the status or locate a fault occurrence, representing hardware including a board or an application.

(*5) Module number: A number identifying a module.

5.5 Class Opt-1 PDU (for Peer-to-Peer Communication)

Figure 41 shows the format of the peer-to-peer communication PDU and setting values for its ADP header.

The data section contains communication data.

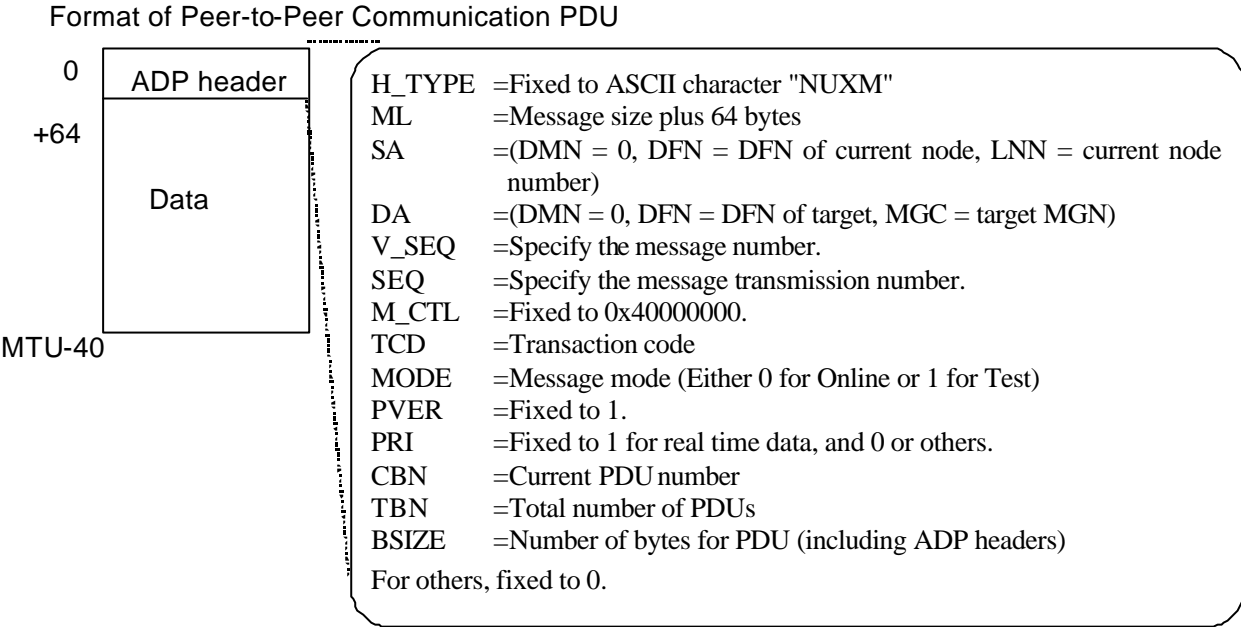


Figure 41 Format and ADP header setting values of Peer-to-Peer Communication PDU

6 Conformance

This section describes conditions required by the specifications on the autonomous decentralized protocol.

6.1 Requirements for conformance

Any devices providing the autonomous decentralized protocol shall implement the communication features satisfying the requirements shown in Table 10 and each communication feature shall implement the functionality/item noted as "Mandatory." Any functionality/item marked as "Optional" may be omitted or implemented in combination as necessary, however, shall be pointed out explicitly if it is implemented.

Table 10 Requirements for conformance against communication feature

Feature	Class	Clause	Requirements for conformance	
			Transmission	Reception
Multicast communication	Base-1	4.1	Mandatory	Mandatory
Alive signal	Base-2	4.2	Mandatory	Optional
Fault information	Opt-2a	4.3	Optional	Optional
Peer-to-peer communication	Opt-3	4.4	Optional	Optional

Table 11 Requirements for conformance against communication-type-specific basic items

Item	Clause	Requirements for conformance					
		Multicast		Peer-to-peer		Alive signal	
		Transmission	Reception	Transmission	Reception	Transmission	Reception
Test support	4.5	Mandatory	Mandatory	Mandatory	Mandatory	Mandatory	Not regulated
Message priority control	4.6	Mandatory	Mandatory	Mandatory	Optional	Mandatory Fixed to 1	Optional
Message numbering management	4.7	Optional	Optional	Mandatory	Mandatory	Optional	Optional
Fragmenting and assembling messages	4.8	Not regulated		Optional		Not regulated	
Maximum message length	—	Mandatory (MTU - 92) bytes		Mandatory 16 KB		Mandatory (MTU - 92) bytes	
Associable arrangement	4.9	Optional		Optional		Not regulated	

Appendix A (Informative) TCD for System

Table 12 System transactions list

TCD	Name	Description
60000 60002	Reserved	Reserved by a special monitoring/control program not complying with the specifications in this document.
60003	Alive signal transaction	TCD transmitted as an alive signal by nodes under data fields and used by other nodes for monitoring.
60004 60007	Reserved	Reserved by a special monitoring/control program not complying with the specifications in this document
60008 60015	Reserved	Reserved for future uses
60016 60059	Reserved	Reserved by a special monitoring/control program not complying with the specifications in this document
60060 65399	Reserved	Reserved for future uses.
65400 65534	Reserved	Reserved for the communication within a controller not complying with the specifications in this document

Appendix B Vendor Code List

A vendor code (BN) identifies a vendor and is contained in al_os_name under the alive signal header.

Table 13 Vendor name and code (informative)

BN	Vendor names
AB	Allen Bradly Japan Co., Ltd.
ABB	ABB Industry.
DS	Daido Signal Co., Ltd.
HI	Hitachi, Ltd.
KS	Kyosan Electric Manufacturing Co., Ltd.
ME	Mitsubishi Electric Corporation
NS	The Nihon Signal Co., Ltd.
OM	OMRON Corporation
OMR	
SE	Seiko Electric Mfg. Co., Ltd.
SS	Seiko Precision Co., Ltd.
TD	Toyo Denki Seizo K.K.
TK	Tokyo Kikai Seisakusho, Ltd.
TO	Toshiba Corporation
YE	Yokogawa Electric Corporation

The Vendor Codes are managed by Manufacturing Science & Technology Center (MSTC). To get a new vendor code, file an application for it to MSTC.

Appendix C (informative) Notes on Implementation

C.1 Message length

The maximum message lengths for multicast communication and peer-to-peer communication are restricted to 1,408 bytes and 16 KB in these specifications, respectively.

NOTE:

The maximum message length for multicast communication may be restricted by the maximum number of the UDP broadcast transmitting/receiving bytes (MTU) for a device.

Maximum message length for multicast communication = MTU - IP/UDP header (28) - ADP header (64)
= 1,408 bytes

Some devices may provide less values than MTU for the maximum number of transfer bytes due to their managing way of build-in buffer, even if MTU for Ethernet is 1,500. When using such devices, the maximum user message length that can be handled with the device shall be clearly demonstrated on the attached operating manuals or guides. For any device that cannot accommodate more than one buffer with the size of 1,408 bytes or more, it is recommended to provide choices for users: 256-, 512- and 1,024-byte buffer size. In this case, also the maximum user message length that can be handled with the device shall be clearly demonstrated on the attached operating manuals or guides.

C.2 TCP connection management

C.2.1 Supplementary control information for connection

If the TCP feature of a node that perform communication has the functionality, it shall be configured.

(1) KEEPALIVE feature

If a node has the KEEPALIVE feature to monitor the state of the target node to prepare for sudden shutdown of the target node due to a crash, the feature shall be enabled previously.

A transmitting node can detect such a fault using a transmitting timeout, while a node only waiting for reception (during a certain period with no transmission) cannot detect the fault. Activating the feature allows such a reception only node to detect an abnormal shutdown of the target node.

(2) Adjusting TCP window size

The TCP window sizes (transmitting/receiving buffer size by connection) of two nodes shall be the same before a TCP connection can be established between the them. If the window sizes between nodes over a TCP connection are different, the communication performance is degraded significantly, so the TCP window size adjusting feature shall be available between nodes. If the target node has a fixed window size, the source node shall adjust its window size to that of the target before connection.

Devices shall clearly describe how many the default window sizes are and the way of adjusting their window sizes in their documentation.

C.2.2 Fault detection for peer-to-peer communication path

Devices shall have features that can detect faulty conditions during peer-to-peer communication, such as a shutdown or fault on the target node, or disconnected TCP due to a LAN fault. If a TCP connection is disengaged, the TCP ports shall be closed and halted indicating a peer-to-peer communication path fault.

For the methods of detecting such a fault, detecting from a LAN communication adapter, TCP transmitting timeout or KEEPALIVE timeout shall be available. An error messaging or error logging feature shall be also available to notify a maintaining person for a fault occurrence.

C.2.3 Other notes

- When a TCP connection is established, the connected nodes shall notify to each other the maximum TCP segment sizes they can handle, and the smaller size is used for the following data transfer. The maximum TCP segment size may vary depending on devices. For example, the TCP transmission/reception unit size between device 1 with 1,350 bytes and device 2 with 1,024 may be 1,024 bytes. A TCP segment contains a message or a packet including an ADP header. The smaller TCP segment size, the poorer the transfer efficiency becomes. The maximum TCP segment shall be clearly demonstrated for any devices.

- The availability of the TCP window size adjustment and the range of size adjustment shall be clearly demonstrated in documentation for any devices.

- A device shall contain its MAC address and IP address during activation, issue an ARP request packet and initialize the ARP tables of other devices (for synchronization to changed MAC address).

C.3 Message priority control

The specifications regulate that the message priority control shall be implemented. When implementing the feature, the following shall be comprised with to prevent any protocol fault with other devices that can occur when a device is connected to a data field running a priority control:

- 1) If the lowest transmitting priority level for a device can cause a problem, users shall be able to configure the priority level for messages transmitted by the device.
- 2) If a user system guarantees that transmitting with priority level 1 is available, transmitting may be fixed to priority level 1.

The priority control is recommended to be implemented based on the above rules (especially item 1) from the viewpoint of the ease of operations). It is recommended to implement transmitting with priority level 0 for default transmission. Of course, the priority level selected by user shall be used if a user define it.

C.4 TCD Access Control

This feature shall be provided to prevent an application from sending or receiving an invalid message by controlling accessing the transmitted or received message from an application. The access control shall be performed by checking TCD. When building the system, a TCD number that is referred to for transmitting/receiving privileges shall be defined for each application. If an application without a required privilege request for a transmission or reception to/from a TCD number, the request shall be rejected as an illegal access.

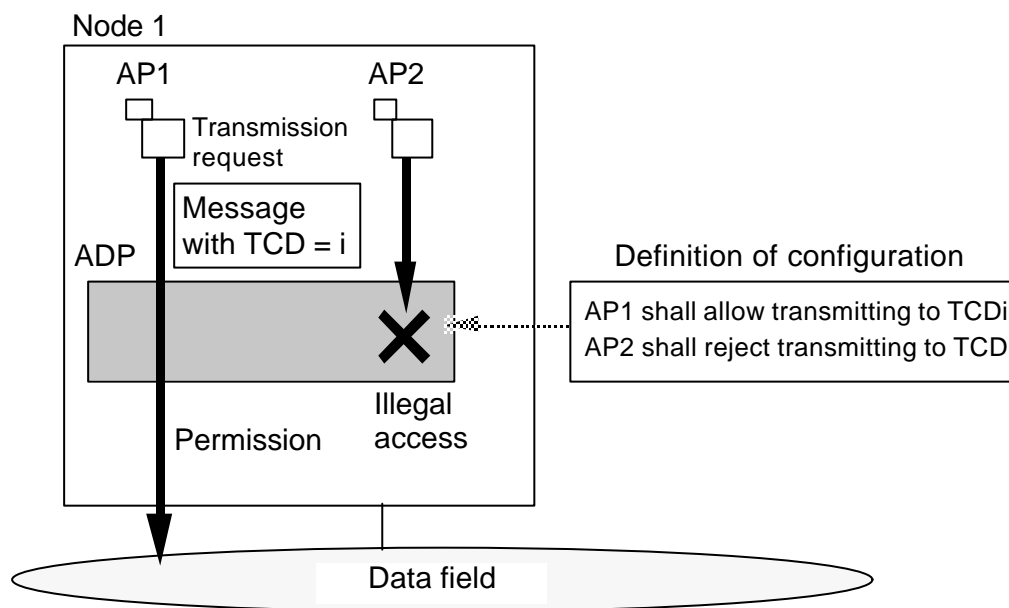


Figure 42 TCD access control

C.5 Logging statistic information for state change in node

A device with the implemented autonomous decentralized protocol shall log statistic information or trace events on state changes including a communication adapter failure or transmitting/receiving buffer overflow during transmission or reception. For a device without the above feature, shall support the communication feature for fault information described in section 4.3, "Class Opt-2-a Feature (for Fault Information Transmission)."

The following are possible state change events within a node:

- Fault of hardware such as a communication adapter
- Activity ration or overflow of a communication buffer
- Fault detection of TCP/IP or UDP/IP protocol

C.6 Extensibility

Nodes or multicast groups shall be able to be added or deleted online to/from a system running on a platform with the implemented autonomous decentralized protocol without interrupting any existing online operation so that the system can easily be expanded in stages.

A device shall be designed to have a capability that allows the device to be added or deleted without interrupting the online system.

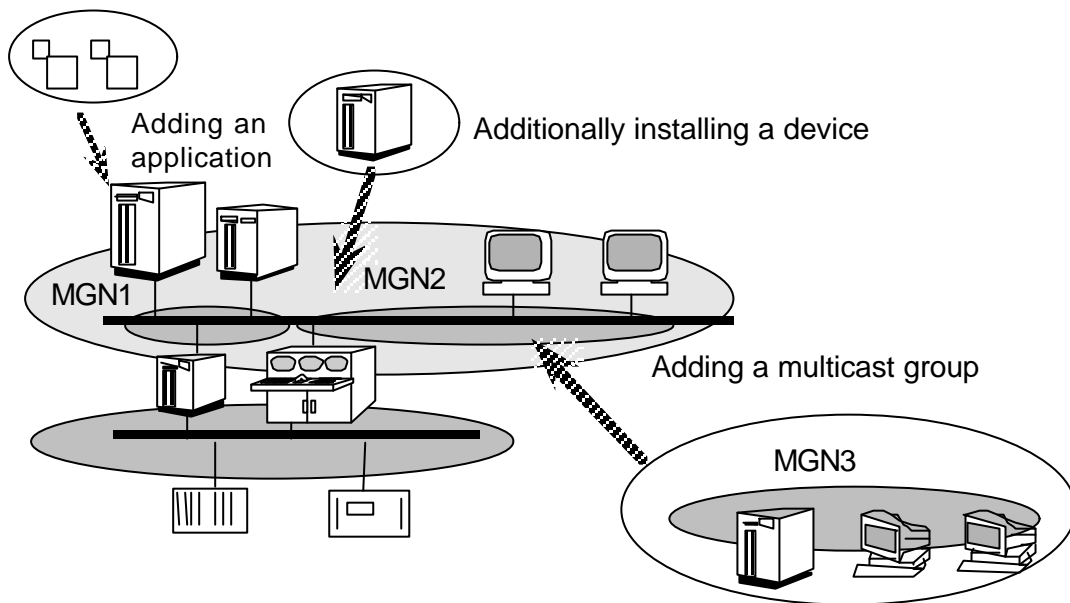


Figure 43 System extensibility

Appendix D (informative) Procedure

The appendix describes sample implementations based on a socket concept used in Windows or UNIX and uses the following API names:

D.1 Transmitting/Receiving multicast communication

D.1.1 Transmitting

The section describes the procedure for transmitting multicast messages using a device with a socket.

- (a) Create the ADP header sent to the destination.
- (b) Add an ADP header to the message, creating PDU.

NOTE:

The specifications do not use message fragmentation, since the message size for the multicast communication is smaller than MTU-92.

- (c) Transmit PDU created in step (b) to the socket for the multicast transmitting port of the target data field by executing "sendto." Specify parameters for "sendto," the broadcast IP address of the target data field and the target UDP port number associated with the target multicast group number.

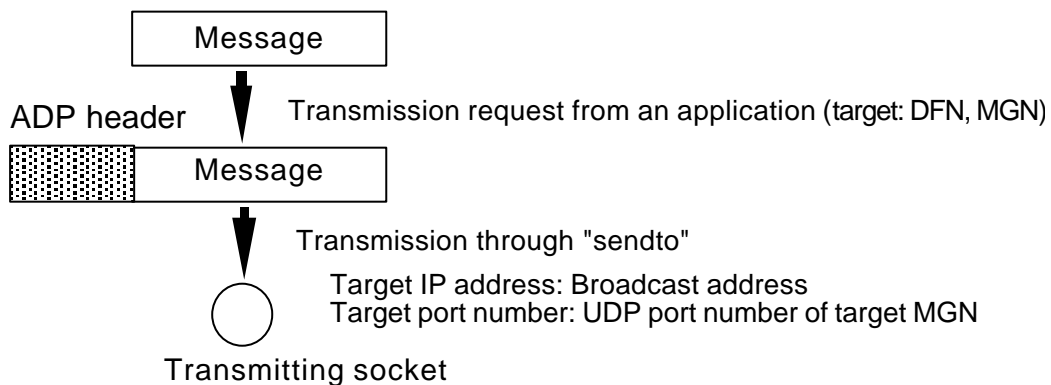


Figure 44 Multicast message transmission

D.1.2 Receiving

The section describes the procedure for receiving multicast messages using a device with a socket.

- (a) Read PDU from a socket for receiving multicast group MGN_i.
- (b) Retrieve the ADP header from the received PDU and check the header validity. At this time, it is required to check the range of valid values in the header to avoid receiving a faulty header from another device that can cause a device fault.
- (c) Pass a message (data section in PDU) to an application based on TCD in the ADP header.

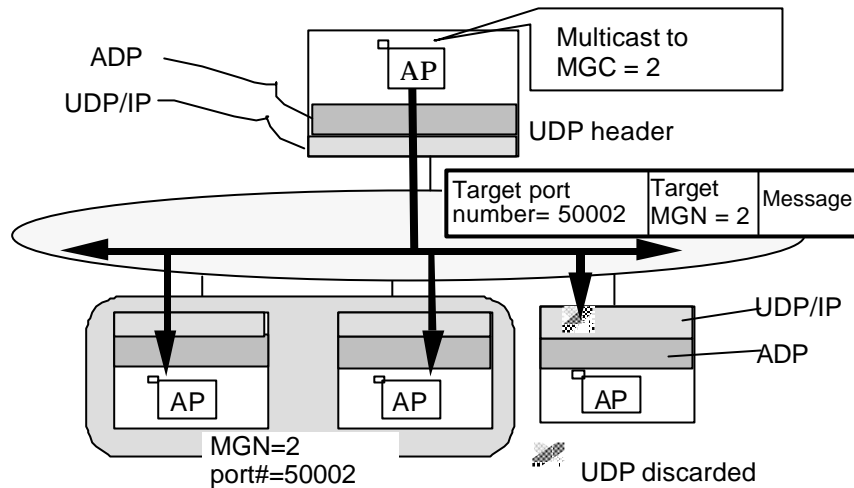


Figure 45 Outline of multicast transmission and reception

D.2 Peer-to-peer transmission and reception

D.2.1 Transmitting

This section describes the procedure of the peer-to-peer communication.

- (a) For a message smaller than (MTU - 104) bytes and that can be sent to LAN as it is, go to step (C). If the message size larger than the limit, divide the message to several packet forms and transmit them following the instruction in step (c). For fragmenting messages, see section D.4, "Dividing and Assembling Messages."
- (b) Add an ADP header to a message (or fragmented message packet), creating PDU.
- (c) Transmit PDU to the connection of the target node. If the window of the target node is closed and transmission is unavailable, wait until the window opens. The configuration must allow reception even when transmission is not allowed, or if there are some preceding packets having requested for transmission, a new PDU must wait (blocked) until all the packets have been transmitted.

NOTE:1

It is recommended before transmitting to TCP, the multicast transmitting socket is made unblocked and transmitting packets is checked if it is available. In this scheme, "write" returns EWOULDBLOCK if transmission using a socket is unavailable. Returning EWOULDBLOCK indicates that other operations are on the way and the target TCP requires the transmission request to wait until the TCP socket is set to "select" and becomes available for transmission. Transmission starts when "select" notifies of transmission availability.

Since TCP transfers data over byte streams, there may be cases where not all the bytes but only several of them are sent responding to an issued PDU transmission request. To avoid data duplication, only the data that have not been sent shall be sent on the second transmission.

NOTE:2

Since the TCP communication shall perform full-duplex communication over one connection, no such an implementation where if either of transmission or reception is made waited and the other is unavailable for processing is allowed.

D.2.2 Receiving

This section describes the procedure of receiving on the peer-to-peer communication.

(a) Reads 64 bytes as the ADP header from the connection of the target node.

Since data communication with TCP is performed over byte streams and it is impossible to clearly know the boundaries of PDUs, users must manage to locate PDU boundaries.

Figures 46 and 47 show the reception procedure. In the former, one PDU is received with one TCP packet, but in the latter received one PDU spans more than one TCP packet.

(b) Retrieves the ADP header from the received PDU to check the header information.

(c) If TBN and CBN of the received ADP header are not equal, that is, the message has been divided into more than one PDUs, steps (a) and (b) are repeated and PDUs are retrieved by the number of fragments to assemble to the original message. For assembling a message, see section D.4, "Dividing and Assembling Messages." If TBN is equal to CBN, it is assumed that the message reception has been completed, so go to step (d).

(d) Pass the message to an application based on TCD in the ADP header.

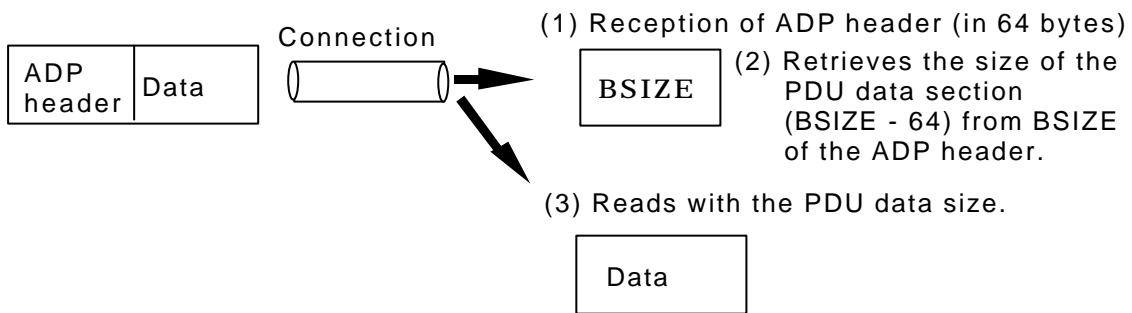


Figure 46 Receiving procedure where one PDU is received with one TCP packet

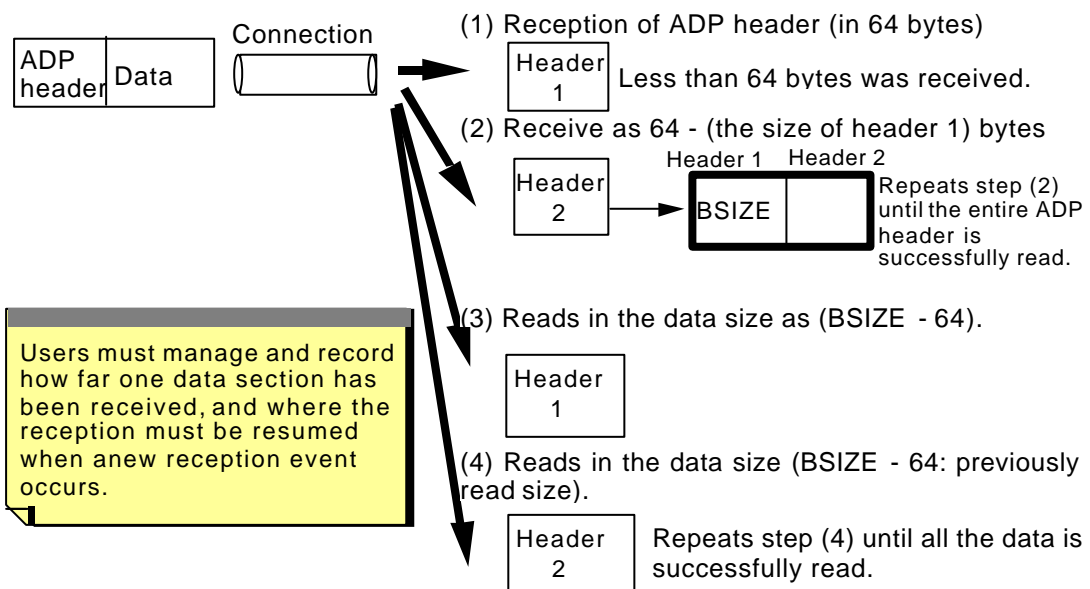


Figure 47 Receiving procedure where one received PDU spans more than one TCP packet

D.3 Transmission number and version number management

D.3.1 For multicast communication

D.3.1.1 Management at transmitting node

Assignment:

- For transmission version management, a version number (V_SEQ) shall be assigned to each node.
- The transmission numbers (S_SEQ) are managed by MGN and by the message priority level (i.e., if there is one level, only one number is assigned).
- The version numbers are assigned with time stamps.

Initialization:

On the system startup and resetting the above transmission, the time of initialization shall be set to the version number V_SEQ and S_SEQ shall be set to 1.

Transmission:

When transmitting a message, V_SEQ and S_SEQ are updated to V_SEQ (version number) and S_SEQ (transmission number) in the ADP header, respectively. S_SEQ will be updated as follows:

- If S_SEQ is not 0x7FFFFFFF, S_SEQ is incremented by one.
- If S_SEQ is 0x7FFFFFFF, S_SEQ is reset to 1 without increment.

NOTE:

If V_SEQ and S_SEQ are set to 0 and 1 respectively, no receiver checks the number.

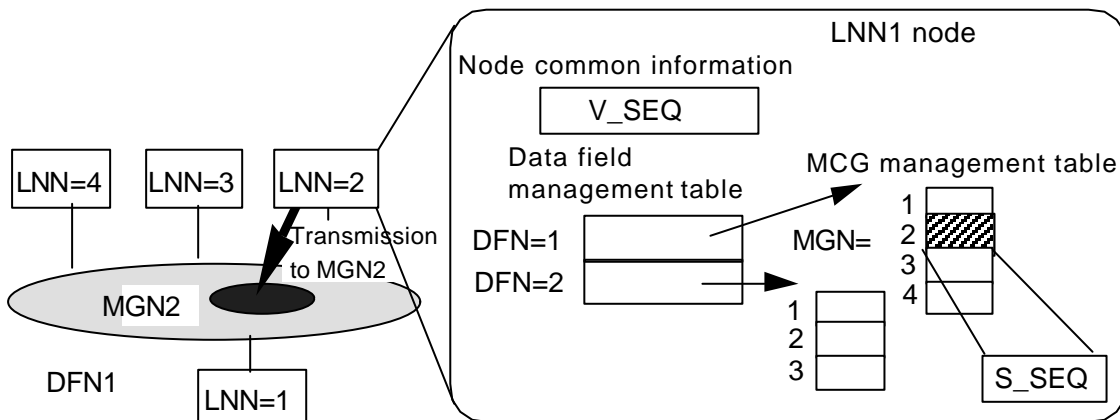


Figure 48 Transmission number management multicast transmitting node

D.3.1.2 Management at receiving node

Requirements:

- Every data field and message priority level shall have a multicast reception number managing table (MCSEQ).
- When receiving a multicast message, the above MCSEQ shall be retrieved through a data field managing table based on DFN and the priority level specified for the source node.
- MCSEQs are arranged as a matrix for each MGN and LNN. For each element of the matrix, the last reception number (R_SEQ) and the last reception version number (R_V_SEQ) are set.
- Individual message source LNN and MGC contain the last reception numbers (R_SEQ) and the last reception version numbers (R_V_SEQ).

Initialization:

For unconditional first message reception after system startup, R_V_SEQ and R_SEQ shall be initialized to "0."

Reception:

When receiving a message, LNN, SEQ (transmission number) and V_SEQ (version number) of the source node shall be read from SA in the ADP header. Requested MCSEQ will be retrieved through the above LNN and received port number. R_V_SEQ and R_SEQ are retrieved from the table for the numbering check described on the next page.

NOTE:

Any message with V_SEQ = 0 and SEQ = 1 is passed to an application without checking the transmission number.

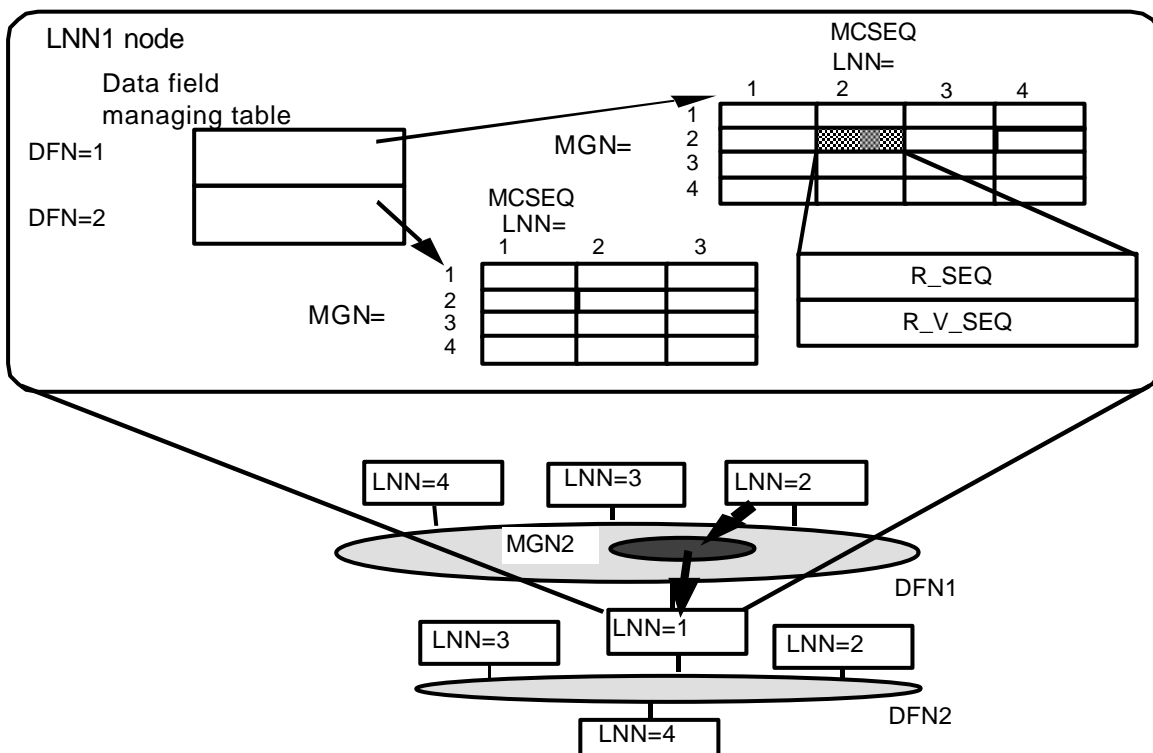


Figure 49 Multicast reception number management

D.3.1.3 Transmission number check for multicast reception

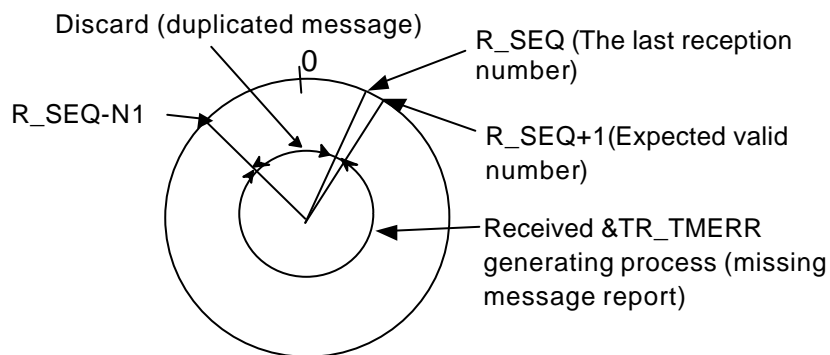
- 1) If R_V_SEQ is equal to 0,
The message is unconditionally received and V_SEQ and SEQ are written into the reception number managing table (MCSEQ).
- 2) If R_V_SEQ is not 0 and V_SEQ is equal to R_V_SEQ,
R_SEQ and SEQ are compared for detecting and deleting a duplicated message if it is found.
- 3) If R_V_SEQ is not 0 and V_SEQ is not equal to R_V_SEQ,
Resetting the transmission number at the transmitting node is assumed, and the R_V_SEQ and R_SEQ values are updated to the V_SEQ and SEQ values.

Table 14 Detailed conditions for number check

No.	Conditions	Determined result
1	$R_SEQ \neq 0x7fffffff \ \&\& \ R_SEQ + 1 = SEQ$	Normal message reception
2	$R_SEQ = 0x7fffffff \ \&\& \ SEQ = 1$	
3	$R_SEQ > N1 \ \&\& \ R_SEQ - N1 < SEQ \leq R_SEQ$	Duplicated message reception
4	$R_SEQ \leq N1 \ \&\& \ (0 < SEQ \leq R_SEQ \ \parallel \ 0x7fffffff - (N1 - R_SEQ) < SEQ \leq 0x7fffffff)$ *	
5	Other conditions	Missing message

N1: Transmission number assumed to be duplicated.

*: For judgement of No. 4, alternatively $0x80000000$ can be added to both R_SEQ and SEQ (to be XR_SEQ and XSEQ, respectively) and converted into the following condition.
 $XR_SEQ - N1 \leq XSEQ \leq XR_SEQ$



N1: Transmission number assumed to be duplicated

Figure 50 Valid reception number

D.3.2 For peer-to-peer communication

D.3.2.1 Management at transmitting node

Assignment:

- For transmission version management, a version number (V_SEQ) shall be assigned to each node.
- The transmission numbers (S_SEQ) are managed by connection for each node.
- The time for the version number shall be set in seconds.

Initialization:

On the system startup and resetting the above transmission, the time of initialization shall be set to the version number V_SEQ, and S_SEQ shall be set to 1.

Transmission:

When transmitting a message, V_SEQ and S_SEQ are updated to V_SEQ (version number) and S_SEQ (transmission number) in the ADP header, respectively. S_SEQ will be updated as follows:

- If S_SEQ is not 0x7FFFFFFF, S_SEQ is incremented by one.
- If S_SEQ is 0x7FFFFFFF, S_SEQ is reset to 1 without increment.

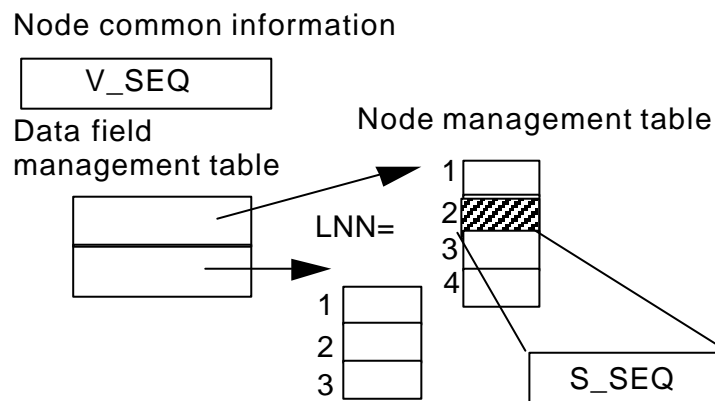


Figure 51 Transmission number management at peer-to-peer transmitting node

D.3.2.2 Management at receiving node

Requirements:

- Every data field shall have a peer-to-peer connection transmission number managing table (PTPSEQ).
- When receiving a peer-to-peer message, the above PTPSEQ shall be retrieved through a data field managing table based on DFN and the priority level specified for the source node.
- PTPSEQs are arranged as a matrix for each LNN. For each element of the matrix, the last reception number (R_SEQ) and the last reception version number (R_V_SEQ) are set.
- Every message source LNN contains the last reception number (R_SEQ) and the last reception version number (R_V_SEQ).

Initialization:

For unconditional first message reception after activation, R_V_SEQ and R_SEQ shall be initialized to "0."

Reception:

When receiving a message, LNN, SEQ (transmission number) and V_SEQ (version number) of the source node shall be read from SA in the ADP header. Requested PTPSEQ shall be retrieved through the above LNN and received port number. R_V_SEQ and R_SEQ shall be retrieved from the table to be processed as described in section D.3.1.3, "Transmission number check for multicast reception."

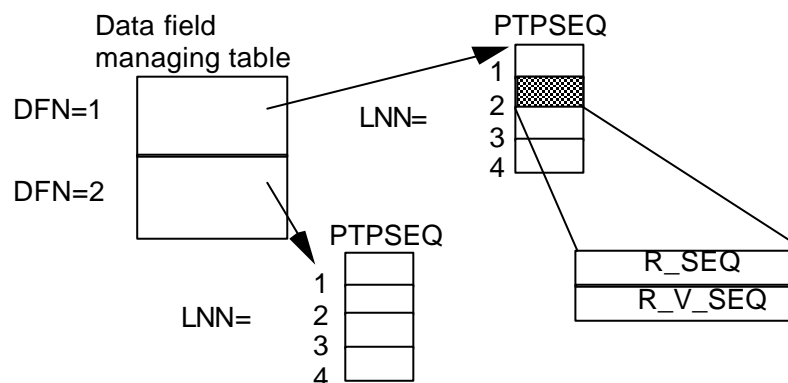


Figure 52 Transmission number and version number management for peer-to-peer receiving node

D.4 Dividing and assembling messages

D.4.1 Dividing and assembling messages

The specifications limits the maximum user message size to 16 KB for the peer-to-peer communication, while, the maximum transfer size (MTU) for LANs is limited to 1,500 bytes under the Ethernet standard. For the peer-to-peer communication, the IP and TCP headers use 20 bytes respectively and the ADP header uses 64 bytes, so available free data storage size can be calculated as (MTU - 104) bytes.

NOTE:

The maximum free user's area in the TCP segment may be less than 1,460 bytes through a negotiation on the TCP segment size when establishing a TCP connection.

If a message size is more than (MTU - 104) bytes, the message shall be divided into multiple PDUs with added ADP headers respectively. The PDUs are sent to the data field in order. The message receiving side receives the PDUs and assembles them into one original message.

Figure 53 shows message fragmenting and assembling processes.

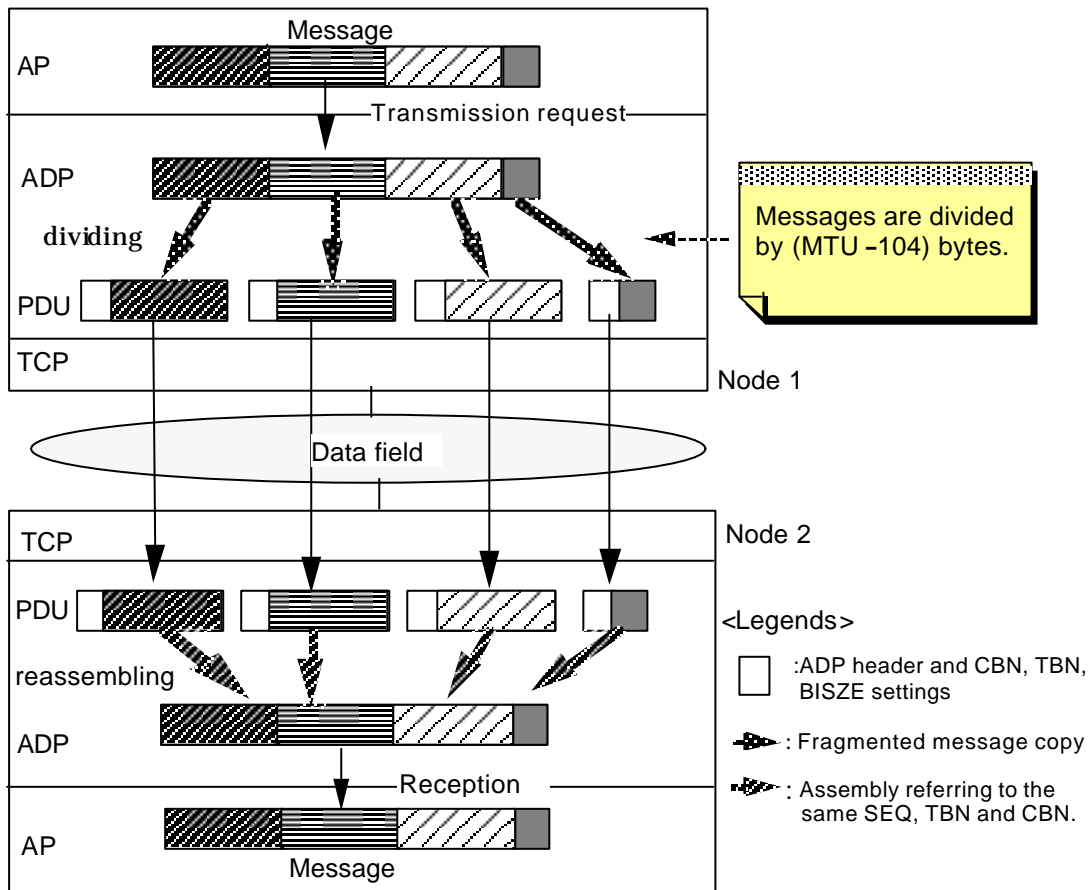


Figure 53 Message fragmenting and reassembling

D.4.2 Header information for fragmentation and assembly

The section describes the header information used for fragmenting and assembling messages. Figure 54 shows the setting examples for the information based on the information on message fragmentation in Figure 53.

CBN: Current fragmented PDU number (1 is assigned to the first PDU number)

TBN: Total number of fragmented PDUs

BSIZE: PDU size

ML: Message size plus 64 (bytes)

SEQ: Message transmission number






Example of fragmentation	Fragmented into four packets				Without fragmentation
SEQ	100	100	100	100	101
CBN/TBN	1/4	2/4	3/4	4/4	1/1
BSIZE	1396	1396	1396	568	1024
ML	4564	4564	4564	4564	1024
Packet					

Figure 54 Sample header information for message fragmentation

D.4.3 Message fragmentation algorithm

The following is the message fragmentation algorithm:

```

UML = User message length
if ( UML > 16384 )
Error handling End
UBS = MTU-104
TBN= (UML+UBS-1) / UBS
if ( UBS < UML ) {
  for (CBN=1, CBN<=TBN,CBN++) {
    if (CBN=TBN )
      HD.BSIZE=UML-UBS*(CBN-1)+64
    else
      HD.BSIZE<-UBS+64
    HD.ML=UML+64
    HD.CBN=CBN, HD.TBN=TBN
  }
}
else{
  HD.ML=UML+64, HD.BSIZE=UML+64
  HD.CBN=1, HD.TBN=1
}

```

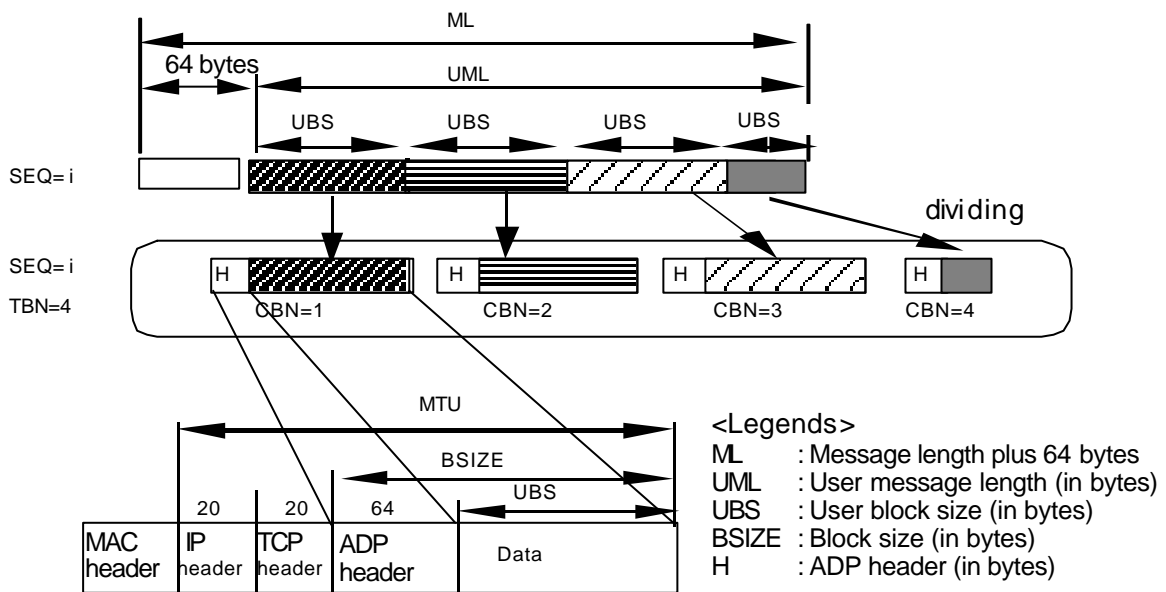


Figure 55 Fragmentation of message to packets

D.4.4 Message assembling algorithm

Before a message can be assembled, the following resources required by the assembling algorithm must be secured and maintained:

Required resources:

- Message receiving buffer

It maintains already received fragmented PDUs until all the remaining fragmented PDUs for a message have been received.

- Fragmented block bitmap (used to check order)

It is used to check which fragmented packet has been received for a message.

- Monitoring timer

The timer monitors the timeout period for each fragmented PDU after reception to avoid keeping the existing fragmented PDUs infinitely after starting to assemble a message and causing buffer space shortage. When the timeout period is reached, the message on the way of assembling shall be discarded. The timeout period shall be 15 sec. in default and adjustable.

The following shows an algorithm where messages are reassembled:

```

Buffer identifier <- Received packet's DFN, TCD, source node number, V_SEQ
If (There is a message being assembled and having the same identifier as the received packet buffer identifier
    && It is older (lower) than the received PDU's SEQ.)
{
    Discarding the message on the way of assembling.
}
else {/** else1 **/
    if (CBN==1 && CBN==TBN) /* Processing received PDU, hereafter.*/
    { Receiving a complete (not fragmented) message }
    else { /** else2 **/
        if (Whether it is a new fragmented message?)
        {
            Queuing list (buffer) waiting for message reassembling is reserved. A bitmap and
            the receiving buffer are reserved for a received fragmented PDU for saving data.
            The monitoring timer starts.
        }
        else{
            The monitoring timer is reset.
            The message assembling queuing list is retrieved referring to the associated
            buffer identifier. A bitmap and the receiving buffer are reserved for a received
            fragmented PDU for saving data.
            if (CBN==TBN && Have all blocks been received?)
            {
                The data section in the list is made available for copying to the user reception area
                End
            }
            else{The monitoring timer restarts, waiting for the next fragment}
        }
    } /** else2 **/
} /** else1 **/

```

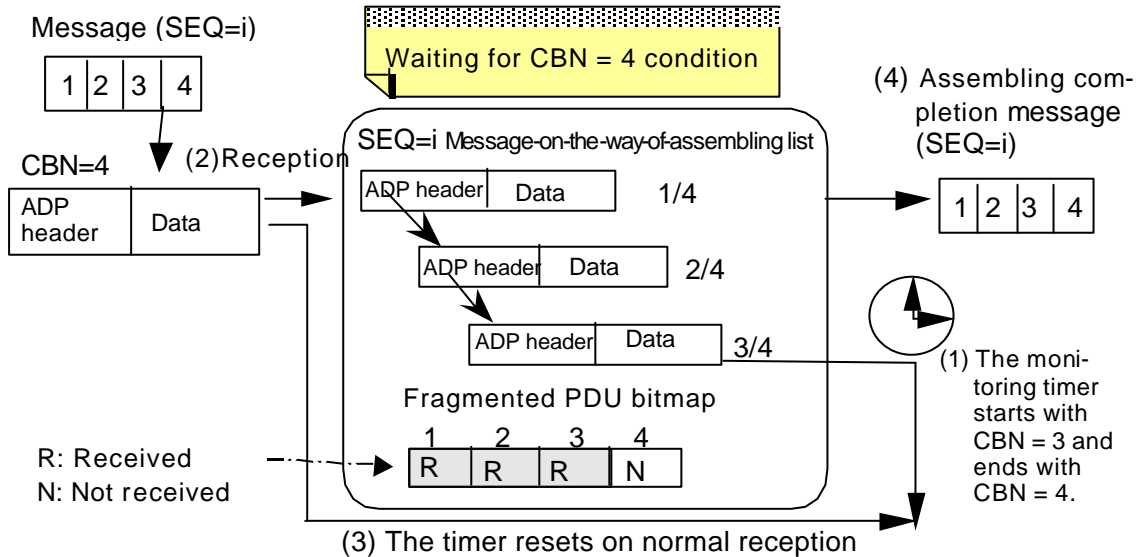


Figure 56 Normal message assembling

If one of the following conditions is met a message is discarded:

- The monitoring timeout period has been elapsed.
- The message with the next transmission number has been reached from the same source node.

The timing of discarding the existing message may be varied depending on the implementation, that is, the existing message may be discarded as soon as the next message reaches or when the timeout period has been elapsed.

Discarding procedure:

- (a) Release the buffer of the message assembling list for which the timeout period has been elapsed.
- (b) Reset the bitmap.
- (c) Notify the user that the message has been discarded.
- (d) The monitoring timer is reset except for the context where discarding occurs at monitoring timeout.

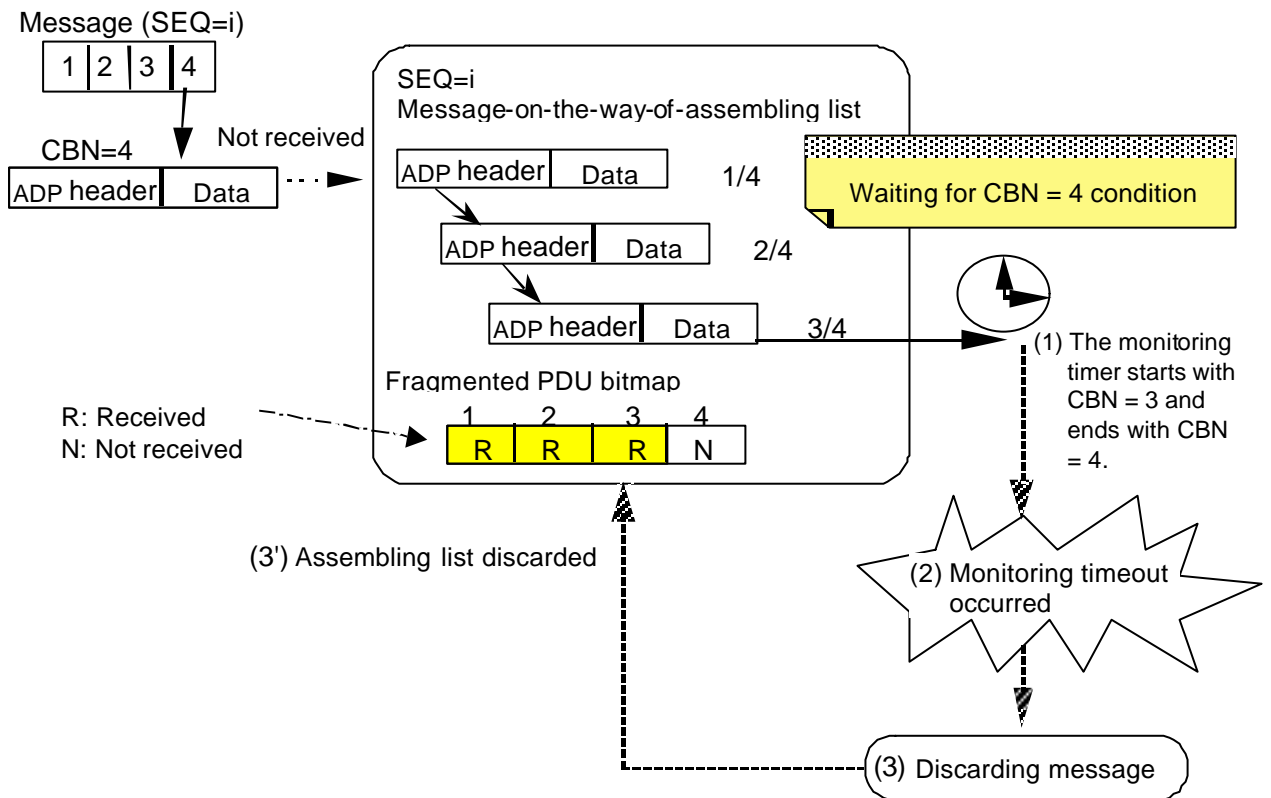
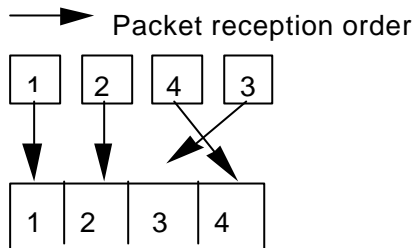


Figure 57 Timeout during message assembling

Figure 58 shows troubles that can be met during message assembling. The implementation shall include remedies against these troubles:

- Case 1: PDU order reversed on the way.

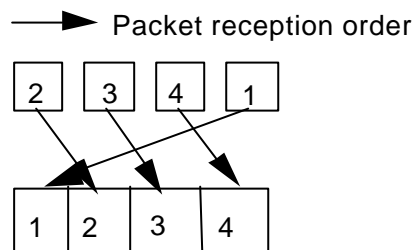


<Legends>

- i : Fragmented PDU number
- X : Missing PDU

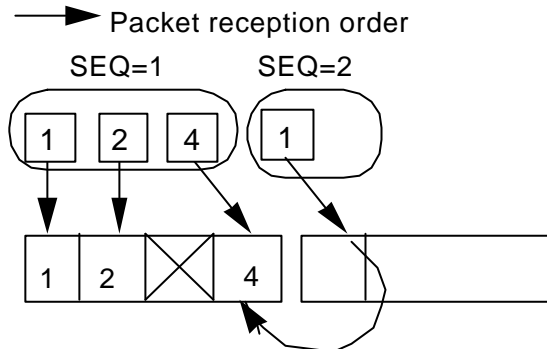
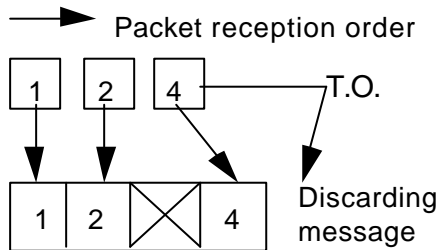
If PDU4 is received after PDU2, the system shall not discard a message immediately but wait until the timeout period of the next packet has been elapsed, because the packet order reversing may be allowed on communications among devices of different types.

- Case 2: Started with reversed PDU order



If a packet reception starts from PDU2, the system shall not discard a message immediately as case 1 is but wait until the timeout period of the next packet has been elapsed. If a packet is missed, the message shall be discarded when timeout is reached or the next message is received.

- Case 3: Detecting missing PDUs on the way



Shall not be discarded at this time but discarded when timeout is reached with SEQ = 1.

When a fault is detected during reassembling a received message, the message shall not be discarded immediately except for the receiving buffer shortage but discarded when message assembling timeout has been elapsed. This timeout period shall be adjustable.

Figure 58 Notes on assembling message

D.5 Node availability monitoring

This section describes monitoring other node status to determine whether it is enabled or disabled based on an alive signal message, for a data field with one LAN configuration.

- 1) When starting to monitor a data field, all the nodes belonging to the data field shall be disabled.
- 2) When an alive signal message is received from a node, the node shall be assumed to be enabled, and the alive signal timeout period (al_tm_out) shall be recorded.
- 3) If the alive signal timeout period (al_tm_out) is elapsed for an enabled node without incoming alive signal messages during the period, the node shall be assumed to be disabled.
- 4) If the alive report mode (al_mode) for an alive signal message is 2 or 3, the node shall be assumed to be disabled and the cause shall be determined whether it is due to a shutdown or maintenance work. Determining the alive report mode (al_mode) shall not be mandatory but may be implemented as necessary.

Example: If the alive signal timeout period (al_tm_out) is 12 sec.,

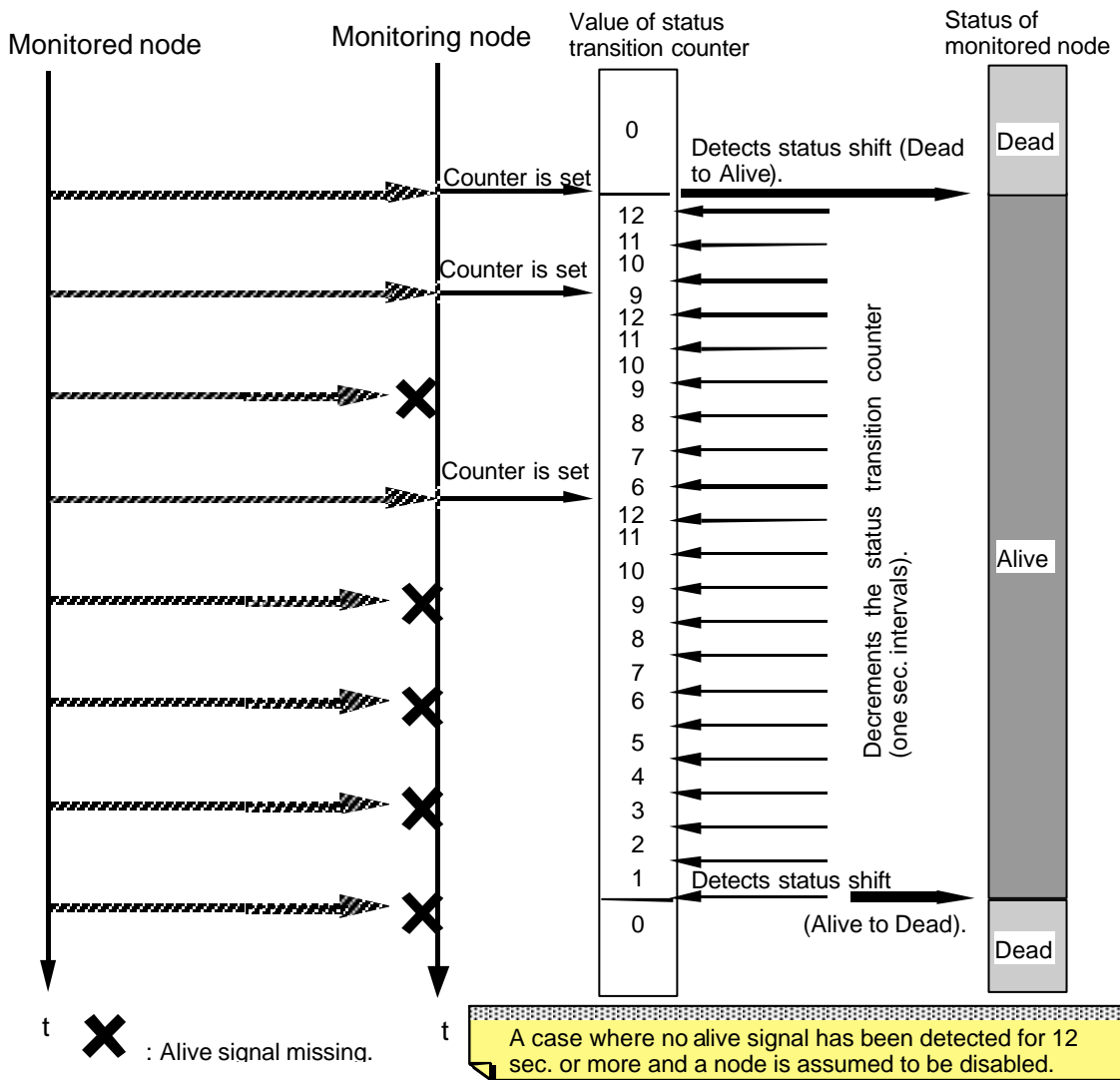


Figure 59 Alive signal monitoring sequence

D.6 Fault information transmission

Before fault information can be sent it must be attached to an alive signal. The following shows how to create fault information, which shall be created when transmitting an alive signal.

(a) Set the number of modules in `al_cnt_mo_alive`, for which the enabled/disabled status must be reported.

(b) Divide the number of modules requiring the enabled/disabled status to be reported by eight (the number of bits for one byte), and secure a storage area (`al_mod_alive`) equal to the number of bytes round off by four (to use long word boundaries).

(c) Set a bit to the reserved storage space (`al_mod_alive`), indicating the enabled/disabled based on the enabled/disabled status of each module.

(d) Register the set of an error number and module causing the error into the list (`al_err_list`) referring to the last fault information transmission (alive signal transmission) and specify the number in `al_cnt_err_list`. The fault (error) information shall be transmitted once or more. To secure the transmission, it is recommended to transmit even an error that has occurred only once (intermittent error) as fault (error) information by the number of times that is calculated by dividing the alive signal timeout interval (`al_tm_out`) with the alive signal transmission intervals. An continuous error shall be transmitted whenever it occurs.

(e) Specify an error name (`al_err_name`). (This is not associated with the transmission number but a name only used to uniquely identify the error number.) The error name is used by monitoring tools to identify a file where error message associated with an error number must be found (see Appendix F.3).

(f) An error code shall consist of two bytes, upper and lower bytes. It is recommended to assign the upper and lower bytes to the summary and detailed codes respectively, as follows:

Two-byte error code:

Upper byte: Summary code

Lower byte: Detailed code

The summary code shall be classified as follows:

0x01: Error of detecting illegal initial data

0x02: Error of detecting faulty header message reception

0x03 or more: Left at the options of vendors.

(g) If any other information is required to be sent additionally as fault information, specify as optional information (`al_cnt_option`, `al_option`).

NOTE:

If the above data size exceeds the available transmission size (MTU - 28) bytes, decrease either of the sizes, `al_mode_alive`, or `al_err_list` or `al_option`. For example, you can achieve this, by dividing an error list into several sub-list for transmission, or eliminating low-priority error from transmission.

Appendix E (informative) Sample Implementation

E.1 Byte order problem

The autonomous decentralized protocol uses big endian for PDU's ADP header byte order, however the data section (message) is not specifically defined. The byte order of the data section may vary depending on the upper section of ADP (i.e., applications).

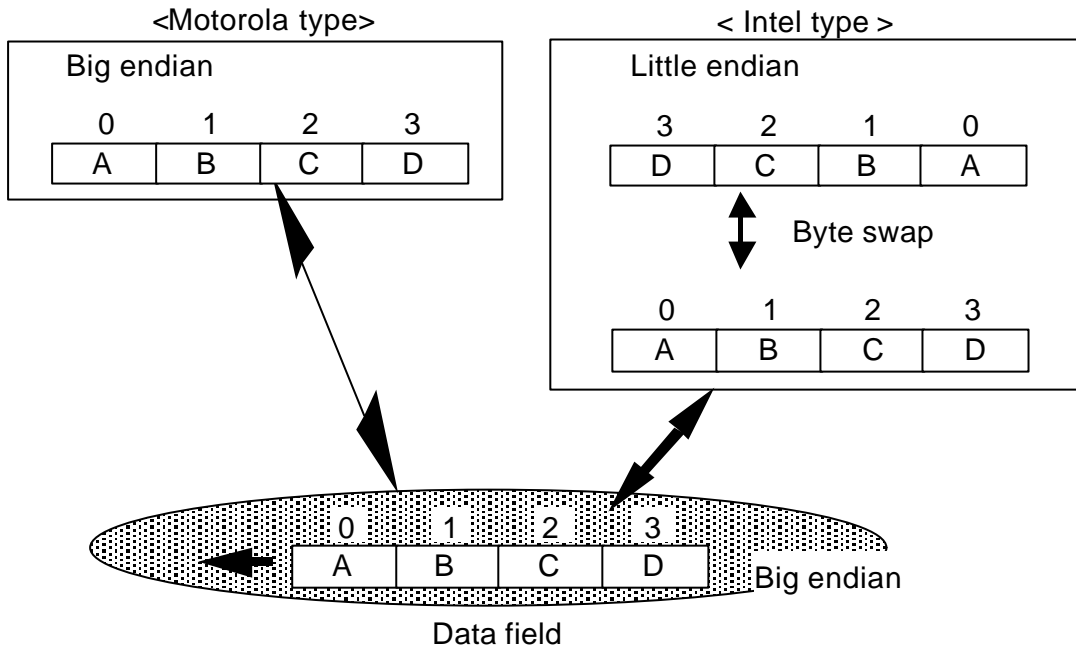


Figure 60 Network byte order

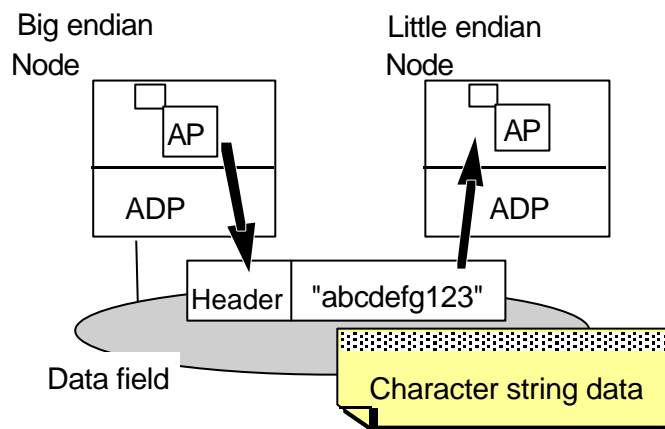
The following methods are recommended for byte ordering of the data section.

- Transfer in character strings

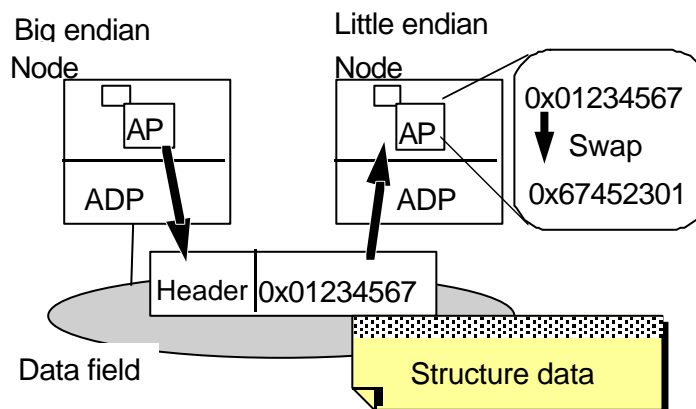
This transfers the entire data section in ASCII character. The applications are not required to perform byte swap nor to regard the byte order type.

- Byte swapping by application

In this method, an application on the little endian node swaps the bytes to convert the byte order on the node with that on the network. Note that for an application with byte order of big endian is on a node or user data is in character strings, no swapping is required.



(a) For character string user data, no byte swapping is required



(b) For structure user data, byte swapping is required

Figure 61 Byte order and byte swapping

E.2 Alignment problem

The autonomous decentralized protocol aligns the header structure in the PDU's ADP header based on the long word alignment to smooth out the alignment discrepancies among different device languages. No alignment problem arises for headers when implementing protocols.

Note that the alignment in the data section (message) is handled by the upper layer (application) of ADP as is the case of the byte order. The following are recommended for the data section alignment:

- Transfer in character strings

This method transfers all the data section in ASCII character strings or the like so that the application is not required to deal with alignment.

- Unifying to the long word alignment by an application

This method defines all the data structures in the long word alignment, allowing using any language alignment when developing applications on nodes. This allows communicating with any devices with different types of alignment, including word, natural or long word alignment.

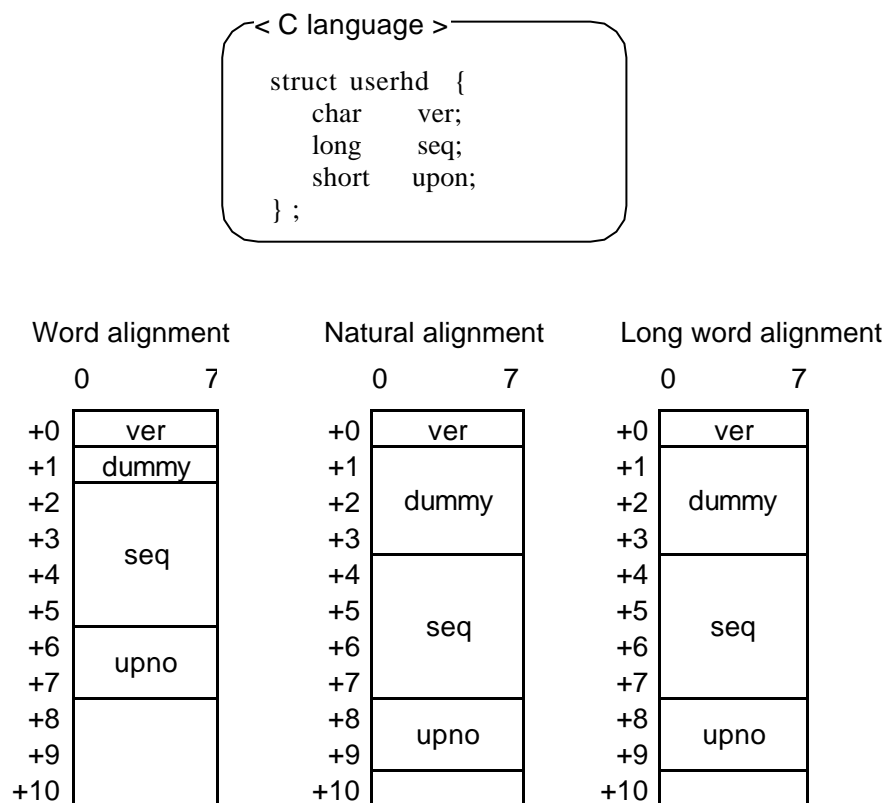


Figure 62 Difference in alignment types

E.3 Alive signal

E.3.1 Conditions

Network address : 128.0
Host address : 128.0.0.1
Data field : 1
Local node number : 2
Local node name : "node2"
Alive-signal transmitting port : 600
Alive-signal transmitting intervals : 10 second

E.3.2 Alive signal message setting values

◆ Target settings

Broadcast address =128.0.255.255
Target port =600

◆ADP header settings

h_type = "NUXM"
ml = 128
sa = 0x00010002
da = 0x00010001
v_seq = 0
seq = 1
m_ctl = 0x80000000 # Multicast
inq_id = 0
tcd = 60003
ver = 0
gtid = 0
mode = 0
pver = 1
pri = 1
cbn = 1
tbn = 1
bsize = ml setting value (128)
fu1 = 0

◆Setting alive signal

al_nd_name = "node2"
al_os_name = "HI_PC_win" # Example :PC from vender H (Windows platform)
al_tm_out = 10 * 4
al_msgserno = 0

```
al_mode = 1
al_protocol = 4
al_chg_time = 0          # Devices not supporting G.M.T.
al_ipaddr[0] = LAN IP address
al_ipaddr[1] = 0
al_ver = 1
al_fu1 = 0
al_fu2 = 0
reserved = 0
```

◆ Alive transmission algorithm

```
While () {
    Sendto ( socketno, alive signal message storage address, size, target sockaddr)
    10 sec. delay
}
```

◆ Setting alive signal for shutdown and transmission example

```
al_mode = 2 # shutdown notice
for other information refer to the normal.
Sendto (socketno, alive signal message storage address, size, target , sockaddr)
```

E.4 Creating Fault Information

Within a node, individual fault points (applications or hardware) shall be assigned with module numbers for identification. Then, a table associating causes of errors that can occur at faulty points with error numbers shall be defined. This table is used to determine the module number and determine the error number associated with the cause or error. ADP implementor shall assign module numbers and associate causes of errors and error numbers.

Each fault can be assigned with detailed data, such as fault register information, as optional information.

Figure 63 shows an example where each application within node 1 is assigned with a module number. In this example, a memory space shortage error occurred at module number 3, the error number was located from the cause of fault, the detailed data on the occurred fault was written into the optional information and the alive signal was transmitted.

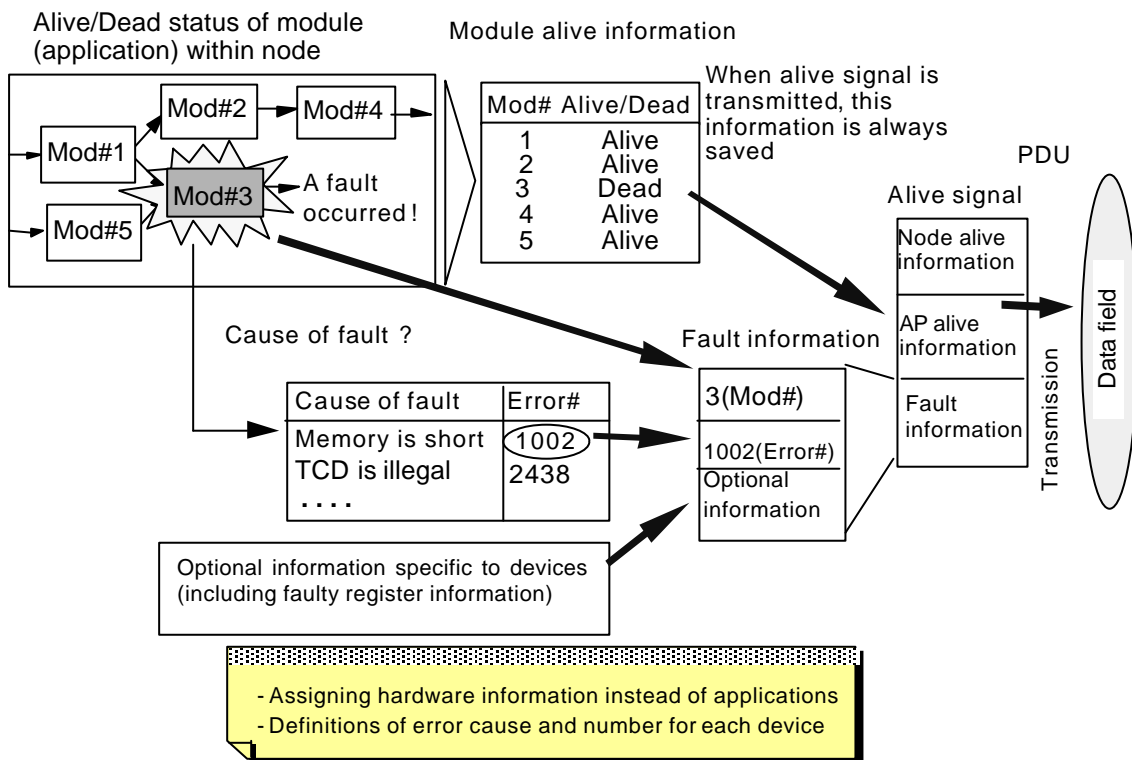


Figure 63 Creating faulty information when a fault occurred

Appendix F (informative) Example

F.1 Displaying node status

The node status within a data field can be monitored using alive signals.

Displaying example in UNIX environment:

- The status of the all nodes within any data field can be displayed in characters.
- The status changes on nodes can be reported to an application online.

Displaying example in Windows environment:

- The enabled/disabled status of the all nodes within any data field can be graphically displayed in different colors.
- The status changes on nodes can be reported to an application online through DDE.

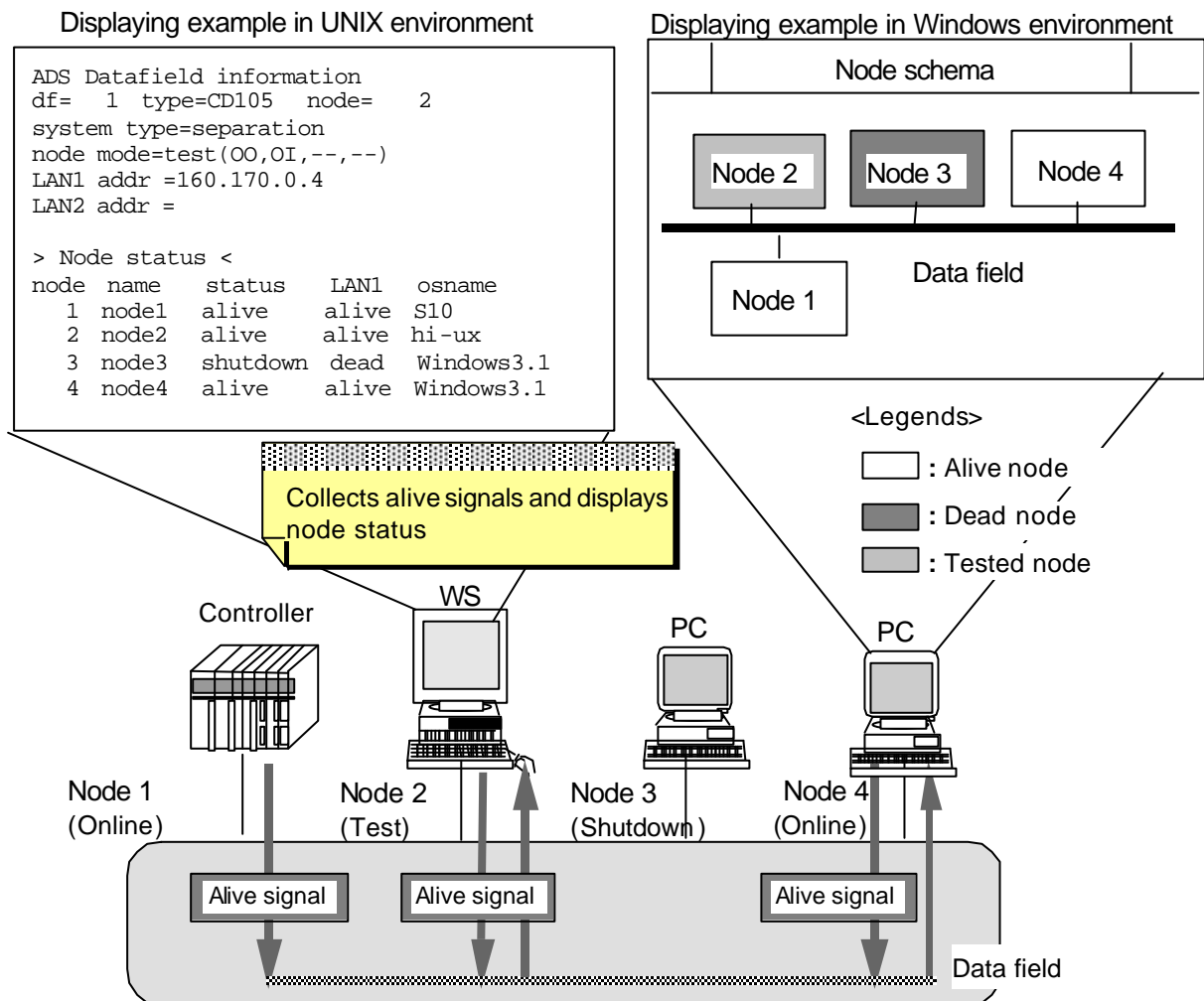


Figure 64 Node enabled/disabled monitoring

F.2 Displaying fault information

Monitoring applications on nodes designated for fault information monitoring can monitor the fault information attached to alive signals. Using the fault information, you can

- Graphically display module structures.
- Display module enabled/disabled status in colors.
- List the occurred errors within the node.
- Log and display the occurred errors within the node.
- Display optional information specific to devices (binary display).

The following diagram shows an example of displaying fault information. In this displaying example, memory shortage occurred in the application, module 3 on node 1, and the status of the application shifted to disabled.

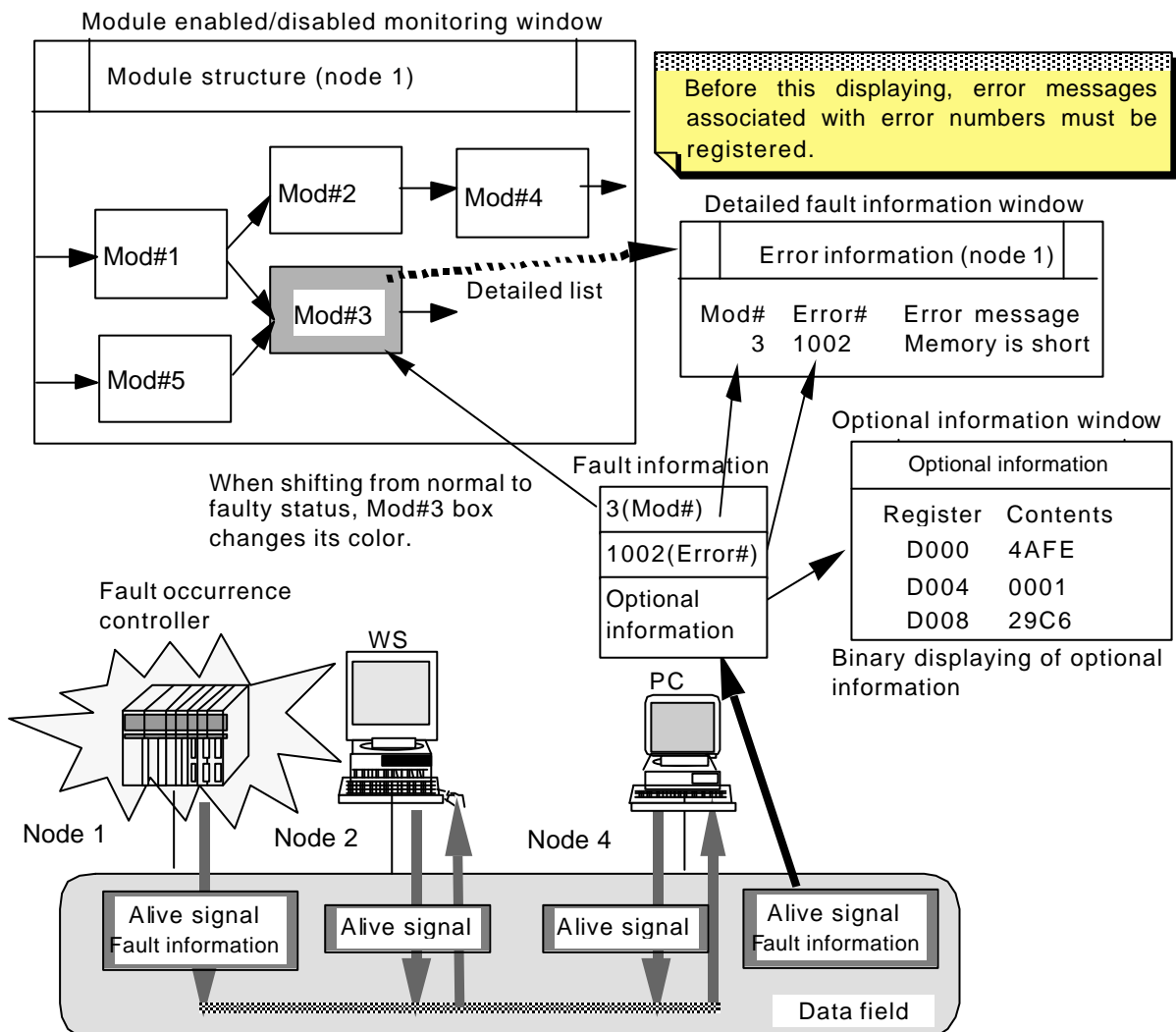


Figure 65 Display example of fault information

F.3 Specifications on error messages for system monitoring tools

A system monitoring tool uses the error message files defined as follows to retrieve an error message associated with an error number (al_err_list) from the received fault information. Devices that transmit fault information shall create error message files associated with fault information and provide it to system monitoring tool developers.

A system monitoring tool shall display an error number and module number as error information, if no file matching the error name (al_err_name) of the fault information transmitted from a node or no error message identified by the error number (al_err_list) is found.

The file shall be stored in a directory specified by a system monitoring tool.

"Vendor-name_Device-name" is an equivalent of al_os_name in the alive signal header.

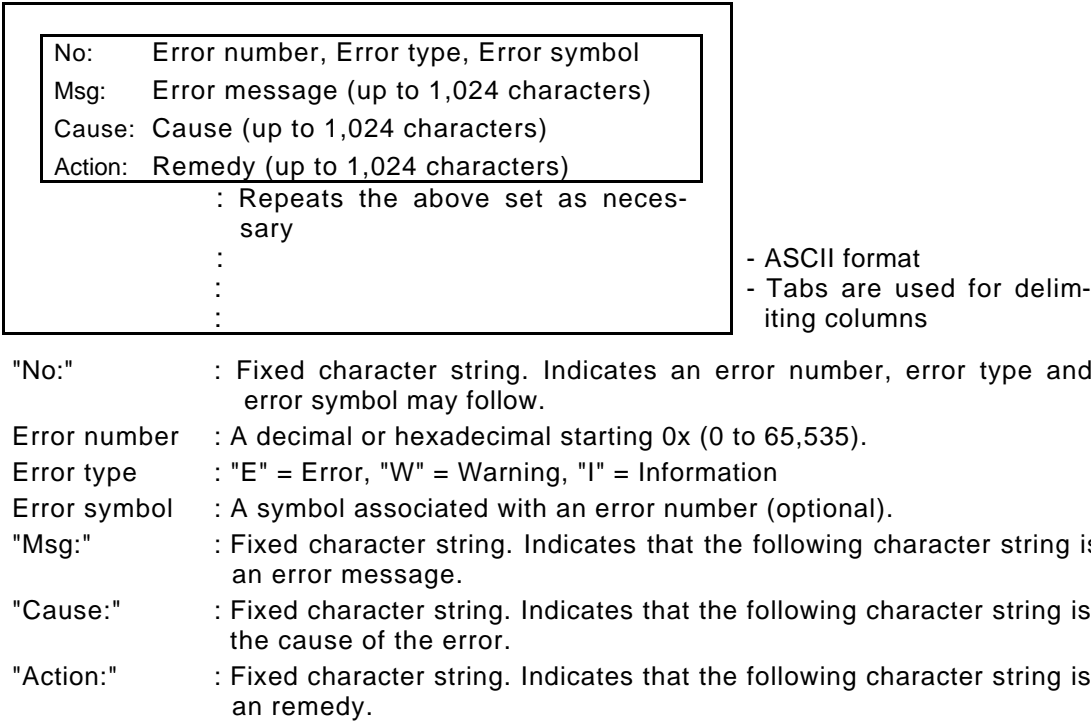


Figure 66 Error message file format

Appendix G (informative) Purpose of Autonomous Distributed System

G.1 Purpose of autonomous decentralized system

The autonomous decentralized system architecture meets wide range of systems, from small to large decentralized systems, and integrates management of different components, such as control servers within systems, workstations, personal computers, controllers and miscellaneous controlling devices, showing synergistic effect by combining different models and cultures.

The autonomous decentralized system architecture was designed aiming the following four main purposes and can deal with the diversification of user needs.

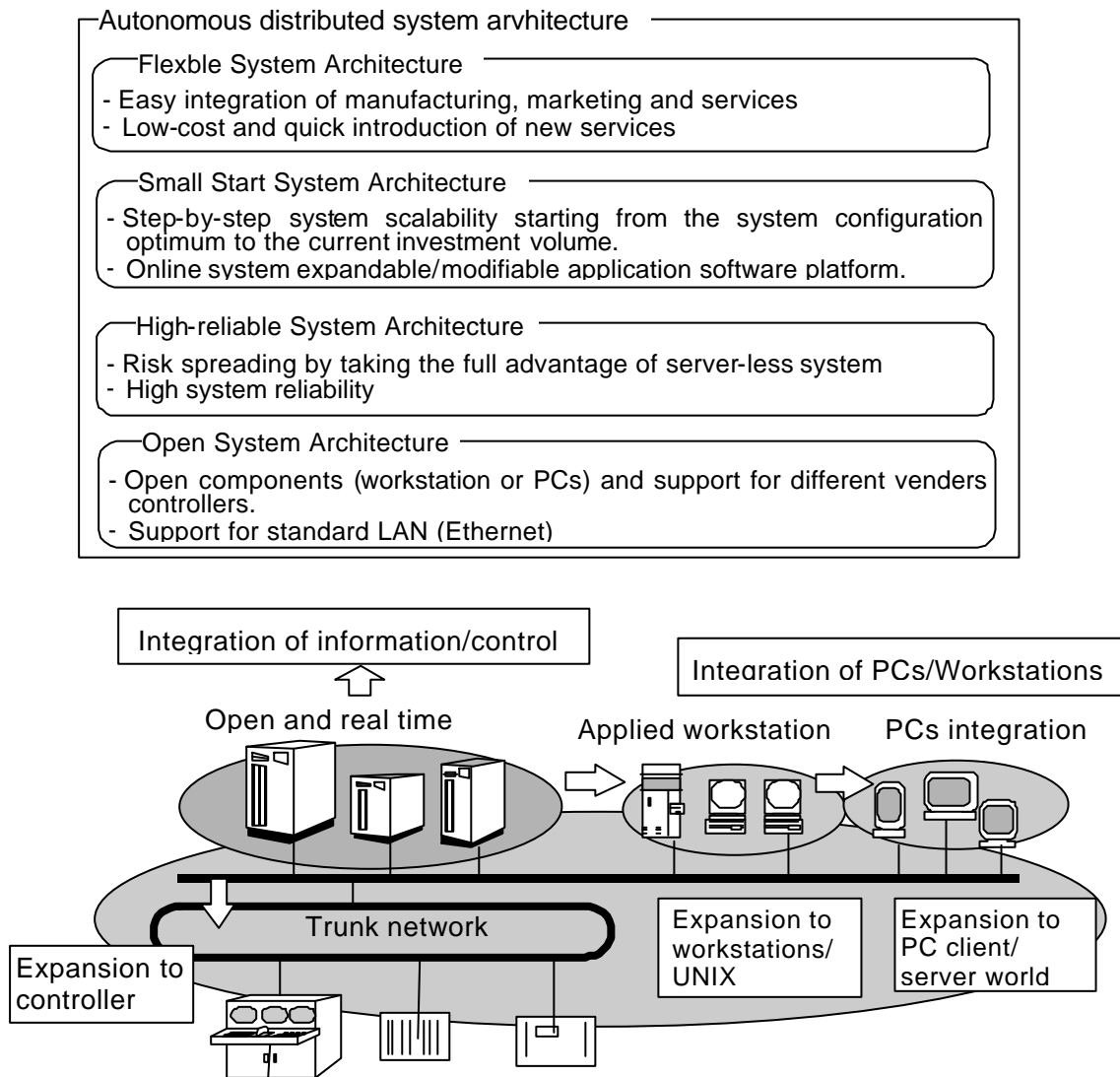


Figure 67 Autonomous decentralized system

G.2 Features of autonomous decentralized system

The autonomous decentralized system defined in the specifications has the following features:

- Rich communication types

A field where data consisting of specific characteristic information and information qualities flows is called "Data Field." Each node multicast data to its upper data field so that other nodes within the same data field can receive the same data. Independence of individual subsystems can be strengthened by dividing a system into more than one data field, and expandability and serviceability can be improved. In addition to the multicast communication feature, peer-to-peer communication feature is also provided between specific nodes, allowing selecting communication features appropriate to communication characteristics between nodes, such as downloading to a specific node.

- Data field management

Using alive signals allows autonomously collecting the status of devices connected to a data field and program or hardware status within devices. The collected data helps managing and controlling the configuration of devices or programs within devices.

- Fault monitoring

When a program or hardware within devices connected to a data field, fault information attached to alive signals allows locating the fault point, determining the cause of the fault and monitoring the detailed information on the fault from a PC.

- Test support

This allows testing the online system without affecting connection.

- Expandability

This allows easy step-by-step system expansion including adding nodes and building user programs online without shutting down the existing online system.

Appendix H (informative) Purposes of Associative Array

H.1 Features

Using associative arrays, a message can be sent with data item names and a data type and other information composing the message, improving self-description of a message.

Data can be retrieved and used by only specifying the set of TCD and data items.

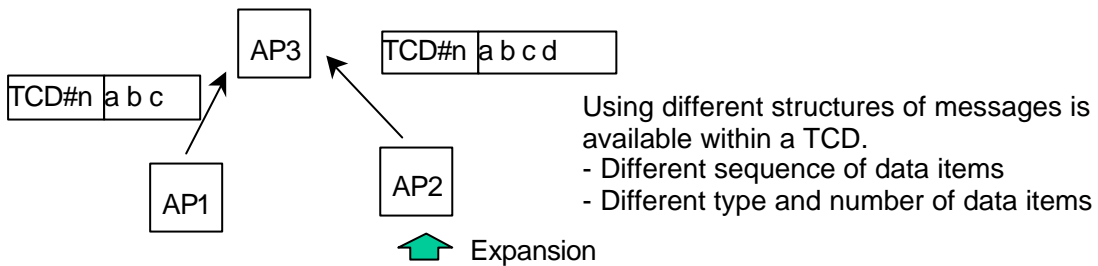


Figure 68 Concept of message for associative array

The following features are available:

- 1) Easy interpretation of messages: Required data items can be retrieved without knowing entire message structure.
- 2) Flexibility
- 3) Modularity: In the above example, given that AP1 and 3 have existed, and if AP3 requires new data "b," only making modification on AP2 and 3 allows operation without modifying AP1. If AP3 has been associated with data "a," "b," "c" and "d," AP1 and 2 can be created in stages (without modifying AP3).
- 4) Online update: When modifying message data structure, it can be updated without interrupting other APs using the message (for adding or sorting data items).
- 5) Decreased TCDs

Data items can be selected for transmission. This allows enlarging the message structure assigned with one TCD, and transmission with less number of TCDs is available.

H.2 Concept of This Implementation Draft

The implementation draft revision 2 proposed in the committee deliberation has been designed in the manner where the conventional type message with fixed data structures and associative array type message with variable data structures can be coexisted within a message to avoid the problems described in the previous chapter. This incurred an opinion that the data section has different categories, the fixed and variable parts, making the protocol complex.

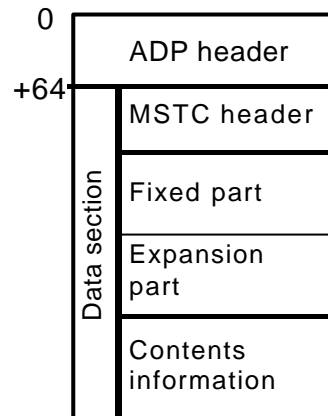


Figure 69 Message structure of associative array implementation revision 2

In this implementation draft, self-explanatory has precedence over other elements, so the conventional type message using fixed structures and the associative array type message using variable structures are separately considered, that is, the MSTC header defines the type of a message and the following data section determines how to interpret it.

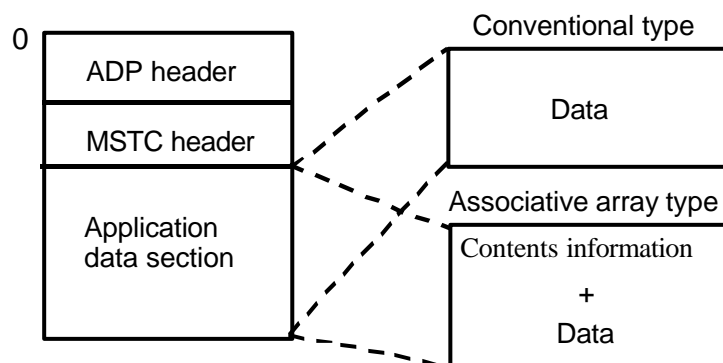


Figure 70 Comparison between conventional and associative array types

When a conventional system without a MSTC header and a system using the protocol must be coexisted, different data fields shall be assigned to them (recommendation) and the application working as a gateway shall convert the protocol.

If the same data field is used, a TCD other than the TCD for existing messages shall be assigned to it and the used protocol shall be determined by TCD.

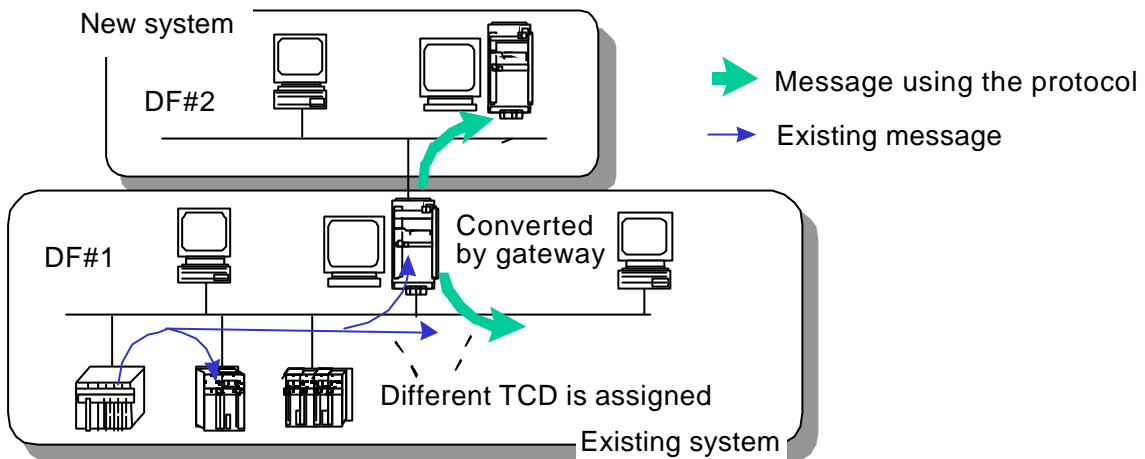


Figure 71 Example of data field separation