Entire Net-Work

Concepts and Facilities

Version 6.3.2

April 2014

ADABAS & NATURAL

This document applies to Entire Net-Work Version 6.3.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: WCPMF-CONCEPTS-632-20191112

Table of Contents

Entire Net-Work Concepts

This document introduces you to Entire Net-Work and provides information about designing your Entire Net-Work configuration.

The Entire Net-Work Concepts document is organized as follows:

Introduction to Entire Net-Work	Explains how Entire Net-Work operates and describes product components and features.
Designing Your Entire Net-Work Configuration	Describes Entire Net-Work design considerations.
Licensing Entire Net-Work	Describes the Entire Net-Work licensing concept.
Starting Entire Net-Work	Describes how Entire Net-Work is started.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to Entire Net-Work

Entire Net-Work for BS2000, z/OS, and z/VSE provides transparent connectivity between client and server programs running on different physical or virtual machines, with potentially different operating systems and hardware architectures. The currently supported set of server programs includes Adabas, Entire System Server, EntireX Communicator, and any other software program that participates in cross-address communications defined by Software AG. A range of client programs are supported, including those written in Natural, the commonly used 4GL provided by Software AG, web-based applications such as Software AG's Jadabas and Tamino, and currently existing Adabas applications.

At its lowest level, Entire Net-Work accepts messages destined for targets or servers on remote systems and delivers them to the appropriate destination. Replies to these requests are then returned to the originating client application, without any change to the application. Entire Net-Work establishes these connections either through its line drivers or, on z/OS systems if the database is UES-enabled, through Entire Net-Work.

The method of operation and the location and operating characteristics of the servers are fully transparent to the user and the client applications. The servers and applications can be located on any node within the system where Entire Net-Work is installed and communicating. The user's view of the network targets and servers is the same as if they were located on the user's local node. Note that due to possible teleprocessing delays, timing of some transactions may vary.

How Entire Net-Work Operates

Entire Net-Work provides transparent support for remote and distributed server processing by supporting the existing Adabas database interface. A user call to Adabas invokes the environment-specific Adabas Link Routine (ADALNK). This routine issues an interregion call to Adabas through the Adabas router (in z/OS and z/VSE, the router is the Adabas SVC). The router, in turn, locates the Adabas nucleus operating in a separate address space or partition, and adds the user call to the Command Queue (CQ). The Adabas nucleus then selects commands from the Command Queue and performs its normal processing.

Because there is no inter-system database communication, this configuration cannot support a user on one system and a database located on one or more other systems. The router was not designed to pass a request across the network to a remote database nucleus or other service on another system.

This section explains how Entire Net-Work solves this problem by simply extending the existing client/server interface:

- Entire Net-Work Establishes the Connections
- Nodes Exchange Information
- The User Issues an Adabas Call
- Entire Net-Work Handles the Call
- Receiving Communicator Accepts the Message

- Reply is Passed to the Requestor

Entire Net-Work Establishes the Connections

Entire Net-Work establishes its connections using two possible mechanisms:

- An Entire Net-Work line driver can be installed on each machine. Line drivers establish a connection with each other through the appropriate access method services. This configuration supports two-way transfer of requests and replies between these two systems, as well as topology and server broadcasting. The line drivers available for Entire Net-Work are: CTCA, DCAM, FCTC, IUCV, SSL, TCP/IP, TCPX, VTAM, and XCF.

  Note: For more information about Encryption for Entire Net-Work (including the SSL line driver), contact your Software AG sales support representative. The documentation for Encryption for Entire Net-Work is delivered separately from the other Entire Net-Work documentation.

- Adabas UES-enabled databases can be connected through Entire Net-Work. For more information, read *Connecting to UES-Enabled Databases through Entire Net-Work*, in the *Entire Net-Work Administration Guide*.
- Adabas UES-enabled databases can be connected through ADATCP -- a direct TCP/IP link to the Adabas nucleus from web-based applications such as Software AG's Jadabas. For more information, read *Connecting to UES-Enabled Adabas Databases* and *Connecting to a UES-Enabled Database through ADATCP* in the *Entire Net-Work Administration Guide*.

Nodes Exchange Information

A node is defined by the Entire Net-Work NODE statement. There is one node per router and at least one router per machine. Each node in a network contains one copy of Entire Net-Work, which monitors any active servers (or targets) on that router and exchanges information with other Entire Net-Work nodes located on the same or on other machines. The Entire Net-Work nodes exchange information about targets accessible to them by means of broadcast messages. This information is maintained in tables by each Entire Net-Work node. Entire Net-Work also identifies itself to the system so that it receives all client requests for locally unavailable servers. Thus it can transport requests to remove servers or determine that a given server is currently not active anywhere. Client programs use server identifiers to direct their requests and are totally insensitive to the actual location of the servers.

The User Issues an Adabas Call

To begin the process, the user issues an Adabas (or any other supported) call. The call is presented to the local router by the Adabas Link Routine, which finds that the requested service is not available locally. Instead of returning Response Code 148 to indicate "service not available", the router passes the call to the locally executing Entire Net-Work communicator.

Entire Net-Work Handles the Call

After accepting the call, Entire Net-Work determines whether the requested service is available on the network. If it is not available, the local Entire Net-Work communicator returns Response Code 148 to the user to indicate "service not available".

If the target is active somewhere on the network, the call (including all required buffers) is packaged as a message and sent across the network to the target node by the line drivers. If necessary and the network topology allows, the message may be routed through many intermediate systems before it reaches its final destination.

Receiving Communicator Accepts the Message

On the target node, the receiving communicator accepts the message and presents the contained call to the locally running service through the router. The actual server regards the Entire Net-Work call as equal to any other call issued from within its own local environment and returns any required reply in the normal manner through the local router.

Reply is Passed to the Requestor

The router passes the reply information to the requestor, which is the local Entire Net-Work communicator, in the same way it passes reply information to a local request. The reply information is then packaged for the return trip to the originating node's communicator, which uses its local router to pass the reply to the user's request back to the client application.

Entire Net-Work Components

Entire Net-Work is installed on each participating host or workstation system requiring client/server capability. The configuration for a given system includes the following components:

- an Entire Net-Work control module;
- control module service routines;
- any required line driver; and
- ADATCP -- a direct TCP/IP link to Adabas UES-enabled databases from web-based applications such as Software AG's Jadabas.

Each system with Entire Net-Work installed and running becomes a *node* in the network. Each node has one or more line drivers that define the node's identity on the machine or host where data can be received. Each line driver has one or more links that contain the driver identifier for sending data to other Entire Net-Work nodes on the same or other machines.

Entire Net-Work Each Entire Net-Work node maintains a *request queue* for incoming requests. This queue is similar to the command queue used by Adabas; it allows the node to receive Adabas calls from locally executing user/client programs, which Entire Net-Work then dequeues and transports to the nodes where the requested services reside.

Each local Entire Net-Work node also keeps track of all active *network services*, and therefore can determine whether the user's request can be satisfied or must be rejected. If the request can be serviced, the message is transmitted; otherwise, Entire Net-Work advises the calling user immediately with Response Code 148, just as the Adabas router would do for a local database request.

Actual network data traffic is controlled by Entire Net-Work *line drivers*, which are interfaces to the supported communications access methods, such as VTAM, IUCV, DCAM, XCF, SSL, TCP/IP, and TCPX, or directly to hardware devices, such as channel-to-channel adapters (CTCAs or FCTCs). Each Entire Net-Work node contains only those line drivers required by the access methods active at that node. In addition, each line driver supports multiple connections to other nodes; this modular line driver design permits easy addition of new access method support to the system.

Summary of Entire Net-Work Features

The following is an overview of Entire Net-Work features:

- **Distributed transaction processing**
  With Software AG's Adabas Transaction Manager, a server for coordinating "two-phase commit processing" in distributed Adabas environments, users query and modify resources under the control of one or more database management systems (DBMSs) operating in one or more system images distributed physically across multiple Entire Net-Work nodes.
- **Operating system-independent architecture**
  Like Adabas, Entire Net-Work consists of many operating system-independent routines, allowing faster and easier adaptation to new operating system versions.
- **Adabas compatibility**
  Entire Net-Work uses Adabas-dependent service routines for the operating system interface as well as for interregion communication, thus avoiding incompatibility.
- **Adabas-like "look and feel"**
  The similarity between Entire Net-Work and Adabas means that the job control statements for running Entire Net-Work are much like those needed to run Adabas. For example, the EXEC statement invokes the ADARUN program for Entire Net-Work just as it does for Adabas, and the ADARUN parameters for Entire Net-Work are a subset of Adabas parameters.

- **Multiple communication access methods supported**
  Access method support is implemented in the form of line drivers. If multiple access methods are needed on a node, supporting line drivers can be added without major reconfiguration of the node itself. Adding a new access method generally requires adding only the line driver module to the library and including the required configuration statements. With this implementation, a client request could be received from a VTAM partner and sent across to another machine using a channel-to-channel or TCP/IP connection for server processing.
- **Access for UES-enabled mainframe databases**
  Adabas UES-enabled databases can be connected through Entire Net-Work. For more information, read *Connecting to UES-Enabled Databases through Entire Net-Work*, in the *Entire Net-Work Administration Guide*.
- **Automatic target status updating**
  All targets and services establishing or terminating communication with the network cause status information to be broadcast to all nodes, eliminating the need to maintain or refer to database or target parameter files at a central location.
- **Unique target ID enforcement**
  Entire Net-Work enforces the Adabas requirement that each enterprise-wide target be assigned a unique target ID. (With Adabas, local targets that are introduced may have non-unique IDs.)
- **Remote processing of client/server request**
  A request can be made from within a Software AG or third party application client program to a server (typically Adabas) located on a remote system, as if the server were running locally with no client changes.
- **One Entire Net-Work per node**
  Allowing only one Entire Net-Work task on each node enforces control over the network topology by maintaining all required information in one place. This avoids confusion in network operation and maintenance. If, however, more than one Entire Net-Work task is required, this can be accomplished by installing additional routers.
- **Single request queue for all remote targets**
  Each Entire Net-Work node maintains only one request queue and one attached buffer pool for economical use of buffer storage.
- **Transmission of relevant buffers only**
  Entire Net-Work eliminates from transmission all buffers that are not required for the particular command. In addition, only those portions of the Record Buffer and ISN Buffer that have actually been filled by the server are returned to the user on a database reply. Software AG still recommends that you set the correct buffer sizes in the control block when coding direct calls; otherwise, unnecessary data may be transmitted.
- **Selectable message blocking and compression**
  Some Entire Net-Work line drivers allow specification of message blocking or message compression. In access methods where a large physical block size is possible, blocking can improve performance during peak message load times by reducing the number of transmissions and requests made to the access method service routines. Similarly, compressing messages reduces

transmission line loading and improves blocking statistics, but increases CPU consumption within Entire Net-Work.

- **All buffer sizes allowed**
  Buffer size support in Entire Net-Work is comparable to that in Adabas, ensuring that all buffer sizes that are valid for Adabas can also be transmitted to remote nodes.
- **Entire Net-Work communication in heterogeneous systems**
  Some line drivers support communication between systems with different hardware architectures, e.g., VTAM and TCP/IP. This allows for client/server communications to and from Entire Net-Work on Windows, OpenVMS, UNIX, and OS/400 operating systems.
- **User exit interface**
  Some Entire Net-Work line drivers support an interface to an optional user exit. The user exit can encrypt or compress data before transmission and decrypt or decompress data after reception. It can decide to accept or reject a connection from a host that is not predefined to Entire Net-Work; when accepting such a connection, it has the ability to select the correct model link parameters.
- **Model Links**
  The "model" link facility allows users to code one or more model links with parameter values that serve as default values for many partners, instead of coding one LINK statement for each partner. As each partner connects, new control blocks are allocated and initialized from the model link.
- **Additional operator commands**
  Entire Net-Work's CTCA, FCTC, IUCV, VTAM, TCPI, TCPX , SSL, and XCF line drivers have the ability to process operator commands that are directed to a specific link or directly to the driver. Some driver and link parameters can be modified with the ALTER operator command while the driver or link is open, thus allowing dynamic reconfiguration of the network. Refer to the specific parameter description for information on possible restrictions about modifying the parameter using the ALTER command.

3 Designing Your Entire Net-Work Configuration

This section describes network design considerations.

Network Design

Network design is critical to providing the best response time to client applications. Client/server applications tend to multiply rapidly, causing networks to expand without following a carefully considered design plan. Entire Net-Work is often installed after a major local or wide area network is already in place.

Whether you are working with a new network or an existing one, it is important to make and follow a plan when installing Entire Net-Work in order to best utilize the existing facilities and provide the best possible response time to your applications.

Entire Net-Work Components

The information in this section refers to the following Entire Net-Work components:

node	An Entire Net-Work instance.
target	A source of information, such as Adabas, EntireX Communicator, or Entire System Server.
client	An application that is accessing a target.

Topologies

Entire Net-Work should be designed to use the same topology used by the underlying network. There are usually several underlying layers of logical and physical configurations that are invisible to Entire Net-Work. Those who design and maintain these underlying layers should be involved in the network design process.

Several topologies are discussed in this section. The bus topology is not included because it does not readily apply to Entire Net-Work configurations.

- Point-to-Point
- Star

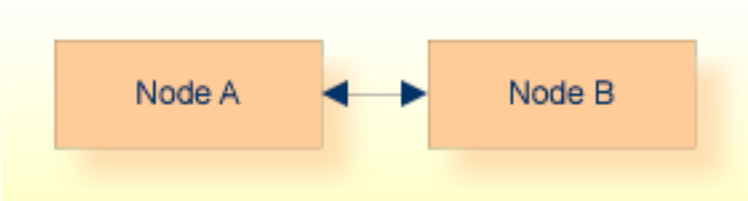
- Ring

Point-to-Point

Point-to-point is the simplest of all configurations. It consists of two nodes connected by a link, and is the way most networks start; for example, a workstation client application using a direct connection to access a mainframe database.

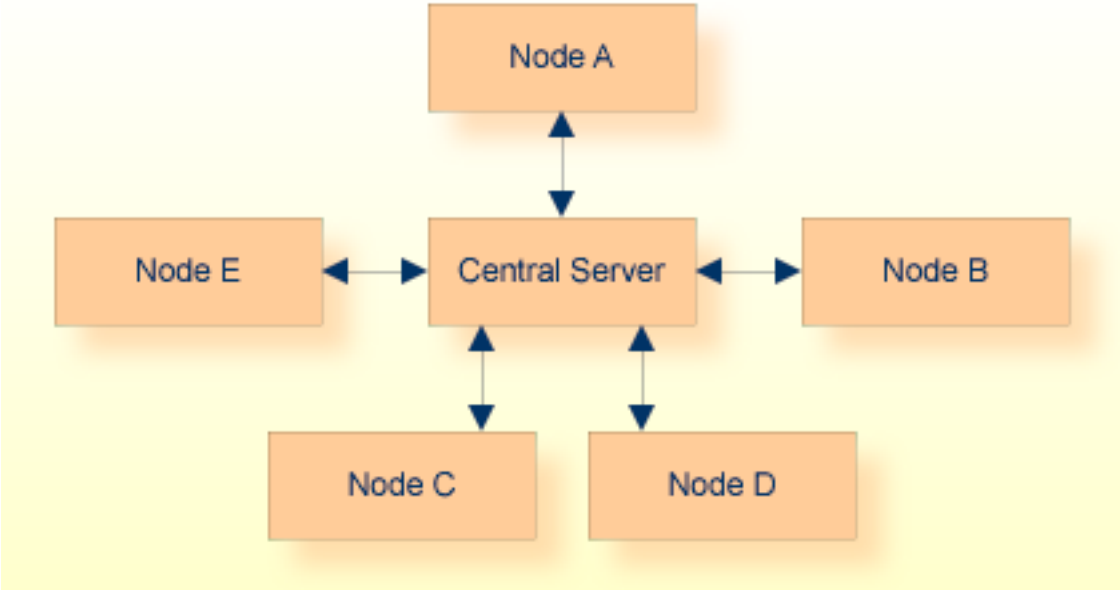
Many modern networks run TCP/IP in a formation that logically consists of many point-to-point sessions. For example, a given client could be attached to FTP://157.189.1.1/etc/file as its server.

The following diagram illustrates the point-to-point configuration:



Star

Point-to-point networks often evolve into Star networks. A central controlling node is attached to each node in the network. The following diagram illustrates the star configuration:



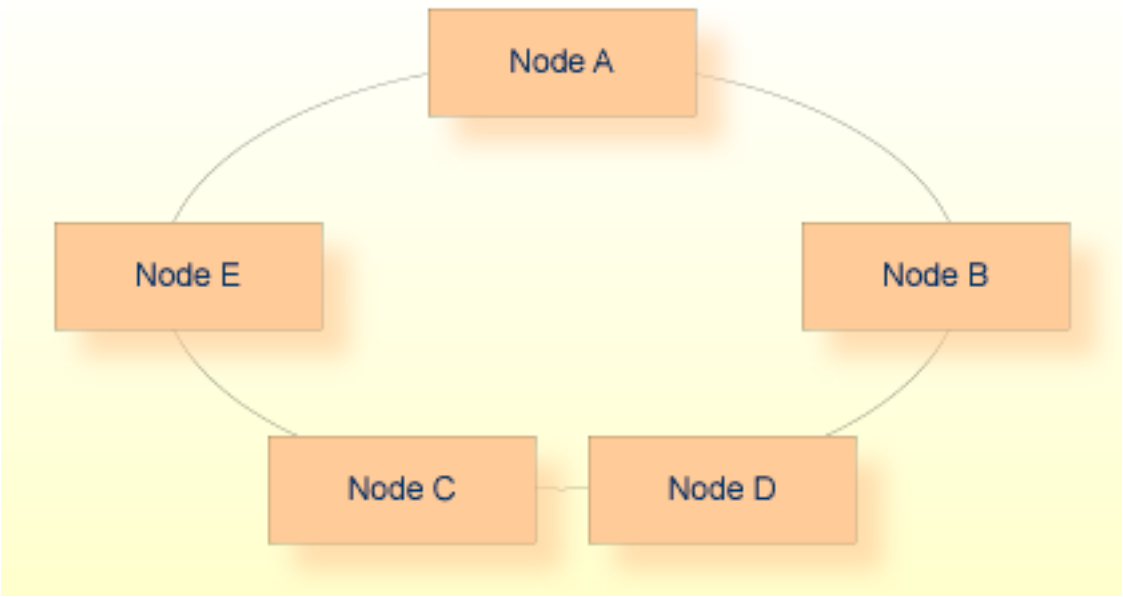
Although targets are normally run on the Central Server, a target running on any attached node is only two hops away from any potential client. The Star configuration is limited by the speed of

the central processor. Its advantage is that it allows central control of the network, which facilitates problem determination.

The original IBM VTAM networks were designed as Star networks, where VTAM itself ran in the central processor and had absolute control over all attached physical and logical units.

Ring

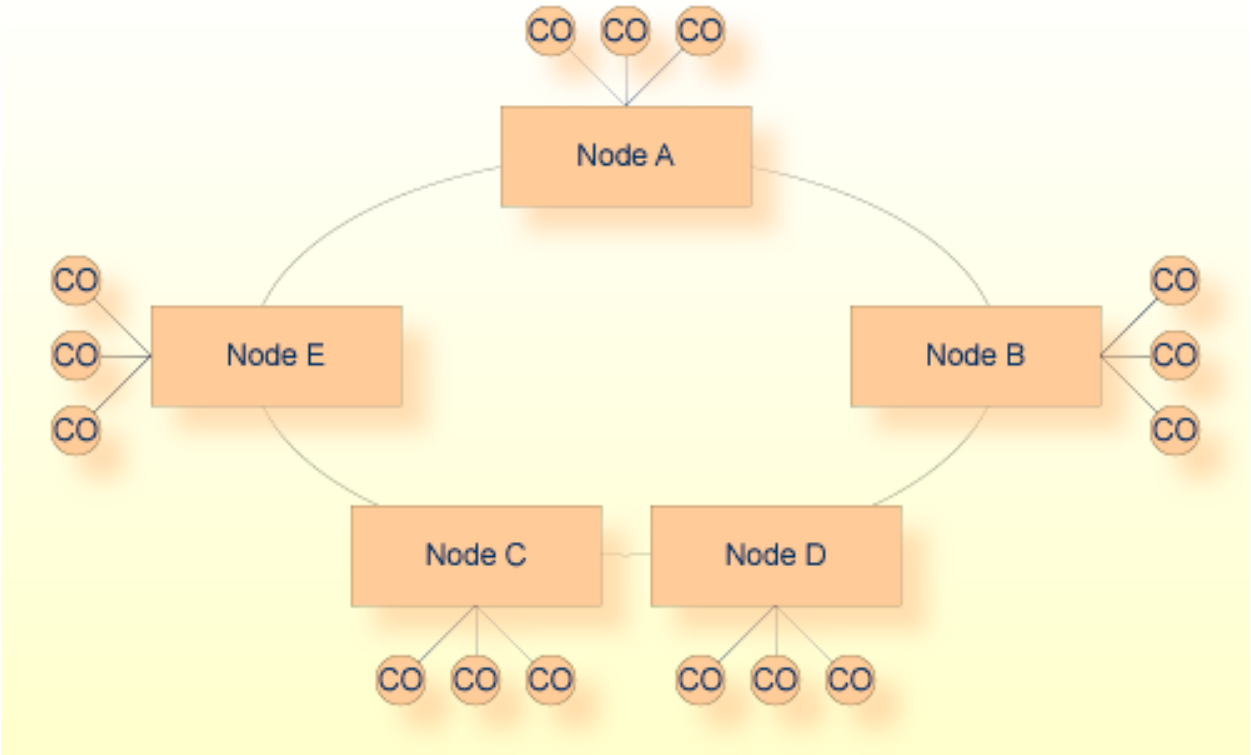
As its name implies, nodes in a Ring network are arranged in a ring, as shown in the following diagram:



In the Ring configuration, it is not apparent which node is the server and which is the client. For this reason, the Ring network is often used among peers, such as Unix servers or mainframes, where clients or servers may run on any node. Because all nodes are peers, problem determination can sometimes be difficult. An advantage of the Ring configuration shown above is that any node is only two hops away from any other node. This configuration also provides **built-in redundancy**.

Expanding the Network Configuration

It is difficult to determine the best way to build your Entire Net-Work configuration. The Star configuration tends to work best in smaller networks, allowing some central control and the ability to route traffic in the most advantageous way. As the network grows, it is important to carefully consider the placement of targets within the network to keep them logically or physically closer to the clients they serve. This may requires a combination of topologies. The configuration in the following diagram, for example, contains a ring of Star networks:

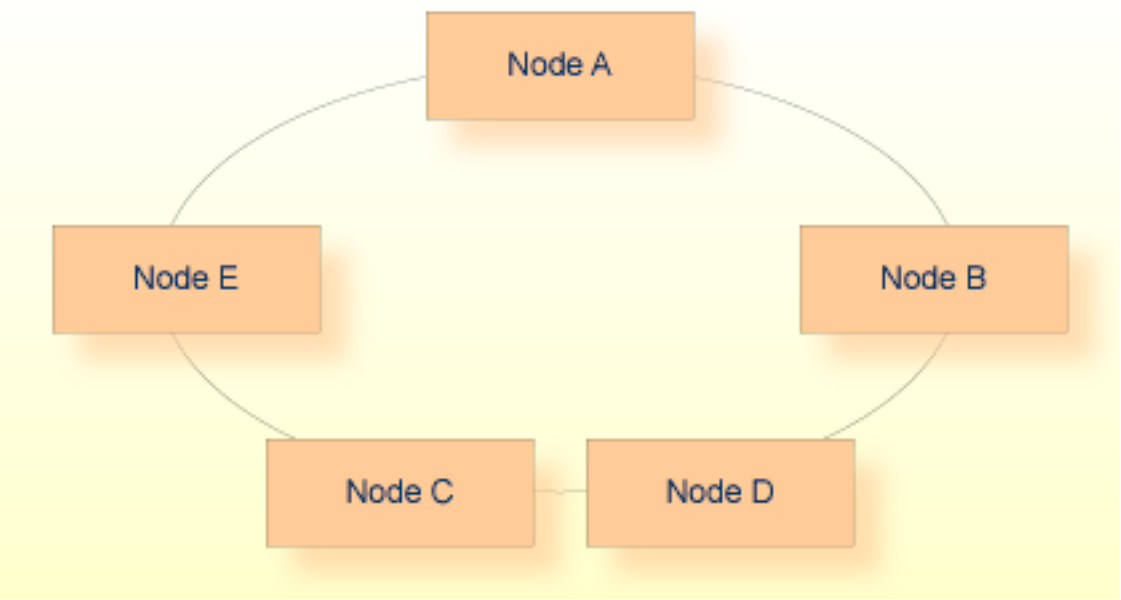


Each server node (Node A, Node B, etc.) acts as the central processor for a group of client-only (CO) nodes, and the server nodes are arranged in a ring formation. None of the client-only nodes is allowed to run a target, so no target can be more than four hops away from any client.

Redundancy

For purposes of this discussion, redundancy means providing multiple paths between nodes. The underlying network may already be redundant. Consult those responsible for maintaining it before implementing redundancy with Entire Net-Work. If implemented incorrectly, the higher level redundancy could defeat the underlying redundancy. Redundancy can be implemented with Entire Net-Work by running heterogeneous links in parallel or by providing multiple paths between nodes via interconnectivity.

Interconnectivity requires that a physical path exist between all points. In the following diagram, for example, traffic can pass in either direction around the ring. If the link between node C and node D is severed, traffic must pass through nodes B, A, and E to connect node D to node C.



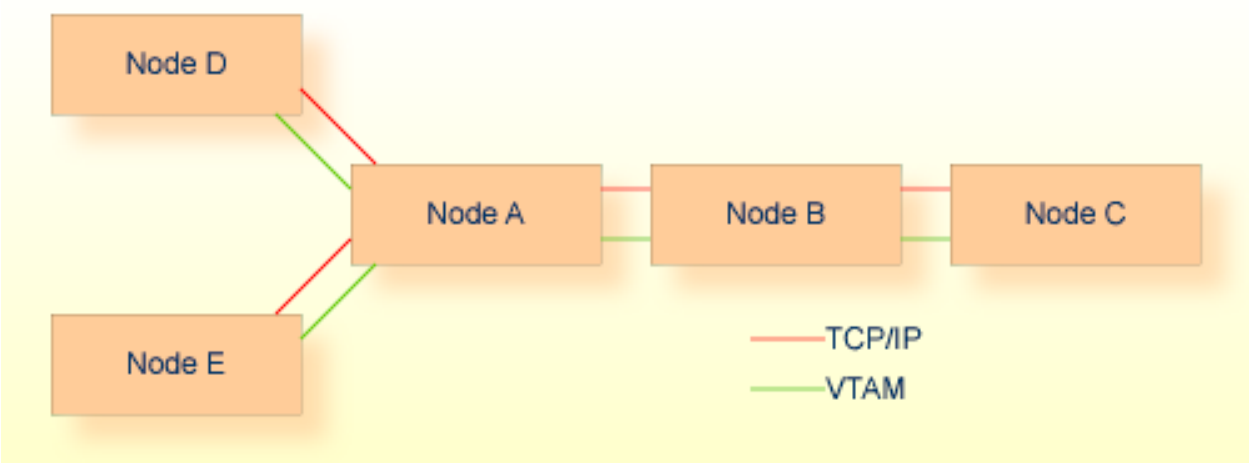
When constructing a logical ring in order to provide redundancy, it is best to determine where the majority of the traffic exists. If most of the traffic moves between node A and node E, for example, putting node D in the middle would add unnecessary processing to the traffic pattern.

Weighting

Weighting is used to prioritize connections. When two or more connections can be used to reach a target node, Entire Net-Work uses the lowest weighted connection available.

Carefully consider weighting the links in your network to provide the most efficient movement of traffic. Assign the lowest weight to the primary link. Assign higher weights to connections that are slower or more expensive; these connections may be used as backup. If the primary connection is severed, traffic can flow over the backup connection. It is important that the connections not share the same physical path.

The example network in the following diagram has both a TCP/IP connection and a VTAM connection between each pair of nodes. If traffic moves at 10 megabits per second on the TCP/IP link and 56,000 bits per second on the VTAM link, the VTAM link should be used as a backup to the TCP/IP link. Giving the TCP/IP link the lower weight ensures the most efficient flow of traffic.



Traffic Considerations

The movement of traffic is the core consideration of network design. The following considerations should be kept in mind when designing an Entire Net-Work Configuration:

- Nodes that share the greatest amount of traffic should be directly connected.
- The lowest weights should be assigned to the links with the highest throughput.
- Maintenance traffic should be taken into account.

Broadcasting

With the exception of client-only nodes, all nodes know about all other nodes in the network and all nodes know about all targets in the network. This is accomplished using broadcast technology.

The first time a node connects to another node in the network, the node that receives the connection sends a message to all the nodes to which it is connected informing them of the availability of the new node. If the new node has others connected to it, then this information is broadcast throughout the network. The newly connected node may also have to broadcast information regarding connected nodes at the other end.

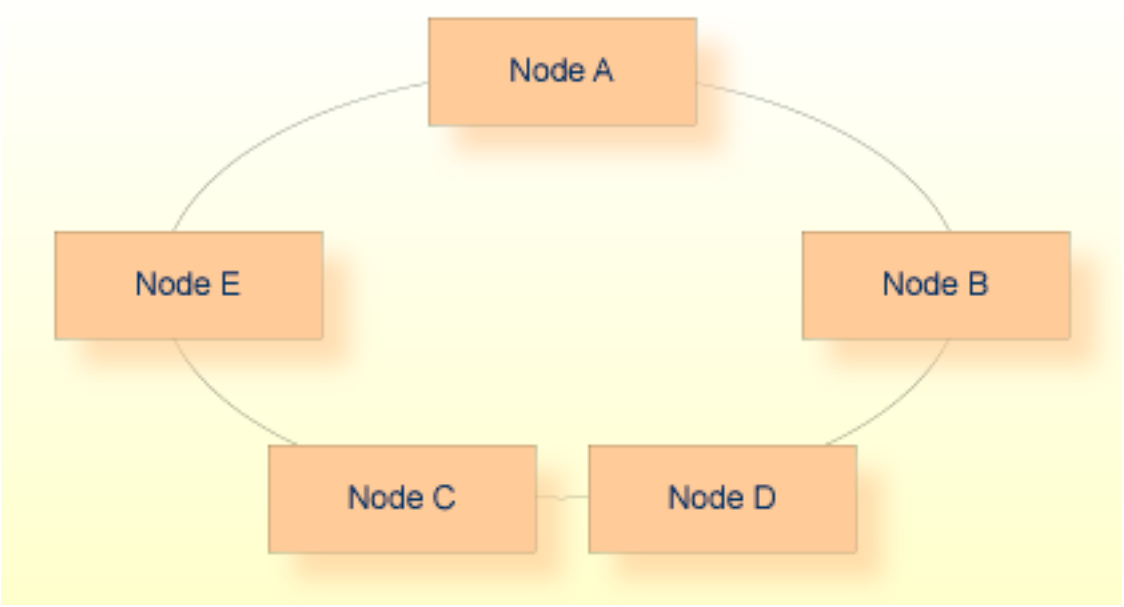
Broadcasting also applies to targets. When a target becomes available or unavailable, the node where it resides will broadcast this information to all the nodes to which it is connected. The broadcast is passed down the line until all nodes in the network are aware of the target's status.

In a large network, broadcasting can create thousands of messages of ever increasing length just to keep every node informed of the status of all known nodes and targets. To prevent wasting

CPU cycles and bandwidth on redundant or unnecessary information, it is important to keep broadcast traffic to a minimum.

There are two good ways to manage this situation:

- Use client-only nodes with the client mode setting turned on.
- Implement redundancy and connectivity in a way that minimizes unnecessary broadcast traffic. Consider the following ring configuration:



In the Ring configuration shown above, it may be tempting to connect node A to nodes C and D, node B to nodes C and E, and so on to provide the most redundant and direct paths possible. Rather than being helpful, however, this creates a great deal of unnecessary broadcast traffic any time a target's status changes on any given node. For example: When a target becomes available on node A, A sends a broadcast to nodes B, C, D, and E. Each of these nodes then rebroadcasts the information to its connected nodes, which in turn creates more broadcast information.

Caution: An overly connected network creates a great deal of undesirable traffic and excessive CPU usage. For this reason, Software AG does not support interconnectivity of more than three nodes.

Client-Only Nodes

Many workstation Entire Net-Work nodes are used only to run clients. Because client-only nodes never contain a running target, information about them need not be broadcast around the network.

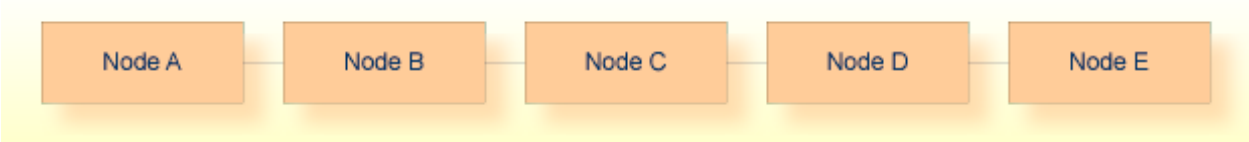
Entire Net-Work for the workstation provides a client mode setting. Client mode tells Entire Net-Work that no information about that node is to be broadcast. If the network has 10 machines that run targets, and 1000 machines that run clients only, the use of the client mode setting clearly results in a large reduction in broadcast traffic.

Number of Hops

A hop is defined as traffic passing from one node to the next. The **point-to-point configuration** shows one hop.

Entire Net-Work must manage information about how to move data between all nodes on the network.

Consider the following example:



In order for node A to get to node E, node A must maintain path information that states that node E is available via node B. It must also, however, maintain path information about all other nodes in the path.

Caution: As a path becomes longer, it becomes increasingly difficult to correctly maintain information about all nodes and paths. For this reason, Software AG does not support more than four hops.

4 Licensing Entire Net-Work

To license Entire Net-Work, use Software AG's standard mainframe licensing code and process, as described in *Software AG Mainframe Product Licensing*, in the *Software AG Mainframe Product Licensing*.

5 Starting Entire Net-Work

Entire Net-Work is started by running a batch job or as a started task. Sample startup (execution) batch jobs can be found in the Entire Net-Work installation instructions for each platform.

Effective with Entire Net-Work 6.3 (and later releases), reusable address space IDs (ASIDs) are supported in z/OS environments. So you can specify the z/OS REUSASID system parameter on the start command for Entire Net-Work. For example:

```
/S NETWORK,REUSASID=YES ↩
```

For more information about the REUSASID system parameter, refer to your z/OS documentation.

Index

W