Author: Harri Sunila

Document: TRS

Date: 15/10/1998

Version: 0.1

TOVE Route Service

Routing in TOVE switch is implemented in TOVE Route Service (TRS) module. The module contains CORBA clients to modify the routing database and query routing information from the database. The CORBA server is implemented using Java programming language and pure Java PSE Pro database.

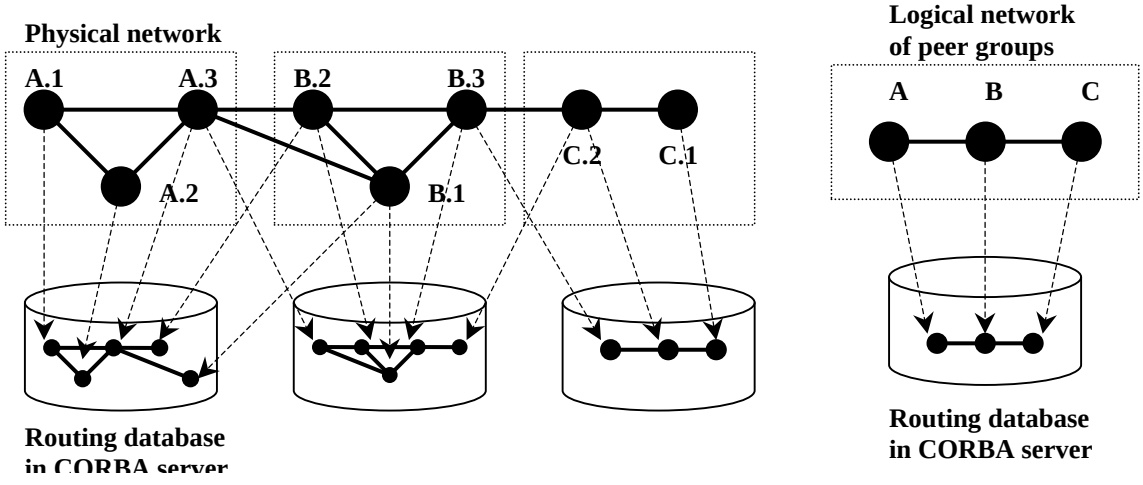
1 Introduction

Routing in TOVE switch is based on PNNI routing [1]. Routing is very much like in PNNI consisting of hierarchical network topology of separate sub networks. TRS module implements a scalable architecture to handle the topology of the network and calculate a number of routes between sub networks or hosts. TRS module does not implement a routing protocol and thus the architecture is suitable for topologically static networks. However a routing protocol may be added afterwards to update the state of the topology database, which makes the software architecture flexible for future development.

2 Architecture

2.1 Architecture of routing databases

TOVE Route Service consists of CORBA server, which is responsible for maintaining the routing database and computing optimal routes between network nodes. On the user side there is separate CORBA clients for both updating the topology database and query routing information. A single CORBA server maintains the database of one sub network. The nodes in a sub network form a peer group and the links within a peer group are called peerlinks. Links between peer groups are called uplinks. The nodes of the peer group as well as peer- and uplinks of the peer group are stored in a single database. The hierarchical routing concept is achieved as in PNNI model. On higher level every peer group is represented as a single node and these nodes form a new peer group. This makes the routing concept very scalable.



The idea of the routing is represented in figure 1 as well as the mapping of network nodes to the routing database.

Figure 1. Mapping of physical and logical networks to routing databases

The routing between distant peer groups is achieved via nested method calls between CORBA servers. Every switch in a network has its own CORBA server, which contains the information of accessible hosts and switches. This means that a single switch is the smallest possible peer group. The same CORBA server is also the access point to the routing information for the switch and all queries of routing information are executed via that server. If the desired routing information can not be found from the first server, the server does a nested method call to the next CORBA server.

2.2 Architecture of routing

Consider the network topology shown in figure 1. Let's assume that the switch A.1 wants to find the route to the switch C.2. It executes a routing query in its private CORBA server (not shown in figure). Because C.2 is not a neighbor of A.1 the server executes a routing query in server containing nodes A.1, A.2 and A.3. Because C.2 is not neighbor of any of these, the server executes a routing query in server containing nodes A, B and C. The whole routing query chain is represented in chart 1.

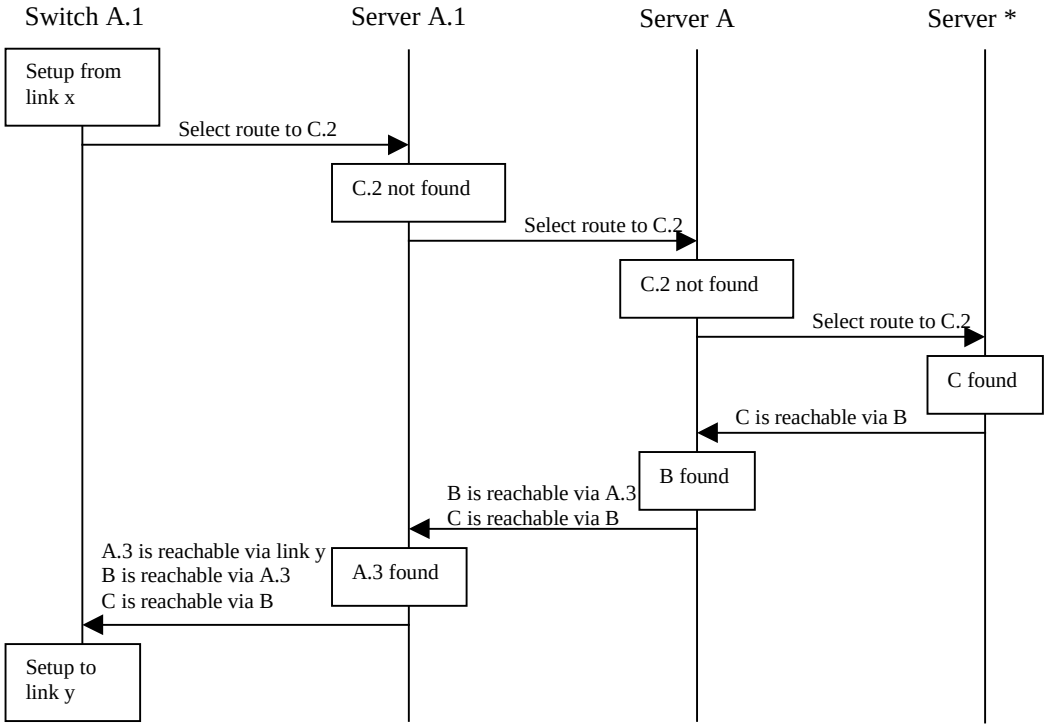


Chart 1. Chain of routing queries

As the previous example shows, routing between peer groups can be considered as partial source routing. The example shows that to reach peer group C.2 the route goes via networks B and C, but the internal topologies of these networks are not visible outside. This means that A.1 can not decide how C is reached from B. TOVE Route Service is similar to PNNI routing model and routing queries generate a list of peer groups or nodes to reach the destination from source. PNNI calls this list 'Designated

Transit List' DTL. Similarly TOVE Route Service provides DTL to the user. The use of DTL helps the routing decision since DTL contains always the next prefix of the route, which can be used in intermediate nodes. If signaling protocols support the use of DTL, the load of routing servers will be minimized since the routing server of the originating node initiates the calculation of the route as far as possible.

2.3 Architecture of Generic Call Admission Control

The function of Generic Call Admission Control GCAC is to assure that the selected route fulfills the QoS demands of the call. TOVE Route Service provides also the GCAC function. The QoS parameters of the links can be stored in the database and the GCAC is done after the calculation of the route is complete. If the route passes the GCAC it is returned to the user, otherwise the routing server tries to find an alternative route.

2.4 Architecture of the software

The following figure and shows how TRS is used by the signaling software. CC is the user of trsRoutingClient and trsManagementClient is used by swSwitch.

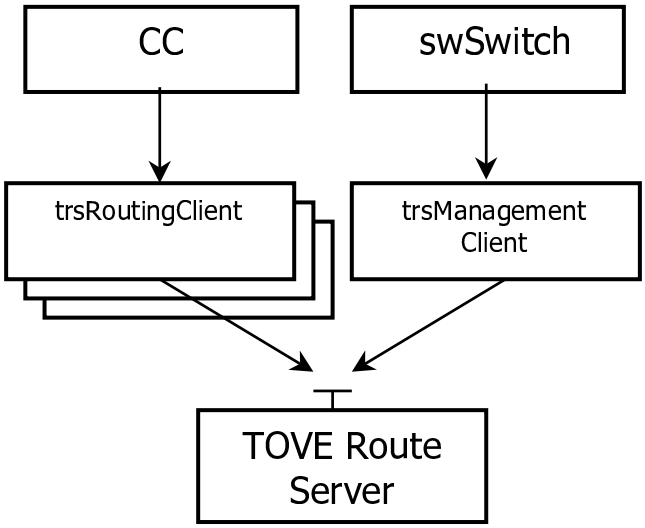


Figure 2. The modules used with TOVE Route Service.

3 Implementation details

The implementation of TOVE Route Service is a straightforward CORBA client-server implementation. IDL operations to modify the database are blocking, but routing information queries are non-blocking operations and results are returned by a separate callback interface. The server serves every operation invocation in a separate thread, which may cause some overhead but simplifies the server implementation.

The routing database is separated from the server implementation and is accessed by a separate DatabaseManager class. DatabaseManager is implemented according the Singleton pattern, which ensures a global access to a single object. DatabaseManager

is responsible of the transaction management and access of database objects including the synchronization of database.

The database is an Object Design's PSE Pro database, which contains a very useful API and class library. The data model of the network consists of separate PersistentNode, PersistentLink and SpanningTreeNode classes, which are persistent and can be stored in the database.

Calculation of routes is separated from the server to a separate thread. An abstract class RoutingAlgorithm implements the Runnable interface and may be run in a separate thread. A derived class SpanningTreeAlgorithm calculates the minimum spanning tree of the network according Dijkstra's algorithm and searches the route from the spanning tree. An alternative routing algorithm may be implemented simply deriving a class from RoutingAlgorithm and implementing the algorithms into corresponding methods.

The client side of TOVE Route Service is divided in two separate classes. The management of the database is performed with class trsManagementClient, which is a wrapper of the management part of the IDL interface. This class performs type translations between OVOPS++ and CORBA types. Routing queries are performed with class trsRoutingClient, which is a wrapper of the routing part of the IDL interface. This class performs similar type translations and offers an interface for the user to make routing queries and the server to make callbacks to CC.

4 Features implemented

The server side provides the client the complete set of management operations. These operations allow the user to modify the database.

- registerNode()

  Add a new node in the database. The node has an unique prefix and a list of links to adjacent nodes.

- unregisterNode()

  Remove a node from the database. This operation removes the node from the database as well as all bidirectional links to this node. If the database contains unidirectional links to the removed nodes, those should be removed beforehand. If the node does not exist, operation throws an exception.

- addLinks()

  Add new links to an existing node. The link contains the information of the reachable node as well as QoS parameters of the link such as cell rate or delay. If the destination node of the link does not exist in the database, it is added. ***Or should we throw an exeption?***

- removeLinks()

Remove some existing links of the node. If the node does not exist, operation throws an exception.

- updateLinks()

  Update the QoS parameters of some node. If the node does not exist, operation throws an exception.

The trsRoutingClient provides the user a method to query routing information. The interface has a single operation to select the route. The operation allows the user to request multiple alternative routes to the destination, but currently only the selection of one route is supported by the server. Currently the server does not perform the generic call admission control procedure either.

5 Known bugs and flaws

There is not known bugs, since TOVE Route Service module has not been properly tested.

6 Future development

In the near future TRS is about to be tested. In the future support of multiple alternative routes as well as the generic call admission control procedure should be added.

7 References

[1] The ATM Forum, *Private Network-Network Interface Specification Version 1.0*, March 1996