Functional Model of a Routing System Architecture

Miguel H. Camelo, Pere Vilà, Lluís Fàbrega
Institut d'Informàtica i Aplicacions
Universitat de Girona
Girona (Spain)
{miguel.camelo|pere.vila|lluis.fabrega}@udg.edu

Dimitri Papadimitriou
Alcatel-Lucent Bell
Antwerp (Belgium)
dimitri.papadimitriou@alcatel-lucent.com

Pedro Pedroso, Davide Careglio
Broadband Communications Research group
Universitat Politècnica de Catalunya
Barcelona (Spain)
{ppedroso|careglio}@ac.upc.edu

***Abstract*—We propose a functional model of the routing system that aims to be used as a common framework from which any specific routing scheme can be derived: the traditional routing schemes OSPF, RIP, BGP, etc., the compact routing schemes, the greedy routing schemes, and new ones. The proposed model will provide a common language to develop routing schemes and will facilitate the comparison between them. The design of a functional model of the routing system is part of the work being done in the FP7 European project EULER (Experimental UpdateLess Evolutive Routing), which aims to investigate new interdomain routing schemes for the Future Internet.**

Keywords- routing architecture; functional decomposition; EFFBD diagrams; Future Internet

I. INTRODUCTION

The main objective of the EULER exploratory research project [1] is to investigate new routing paradigms so as to design, develop, and validate experimentally a distributed and dynamic routing scheme suitable for the future Internet and its evolution. The resulting routing scheme(s) is/are intended to address the fundamental limits of current stretch-1 shortest-path routing in terms of routing table scalability but also topology and policy dynamics (perform efficiently under dynamic network conditions). Therefore, this project will investigate trade-offs between routing table size (to enhance scalability), routing scheme stretch (to ensure routing quality) and communication cost (to efficiently and timely react to various failures). The driving idea of this research project is to make use of the structural and statistical properties of the Internet topology (some of which are hidden), as well as the stability and convergence properties of the Internet policy, in order to specialize the design of a distributed routing scheme known to perform efficiently under dynamic network and policy conditions when these properties are met.

The project will develop new models and tools to exhaustively analyze the Internet topology, to accurately and reliably measure its properties, and to precisely characterize its evolution. These models, that will better reflect the network and its policy dynamics, will be used to derive useful properties and metrics for the routing schemes and provide relevant experimental scenarios. The project will develop appropriate tools to evaluate the performance of the proposed routing schemes on large-scale topologies (order of 10k nodes). Prototype of the routing protocols as well as their functional validation and performance benchmarking on the iLab.t experimental facility [2] and/or virtual experimental facilities such as OFELIA [3] will allow validating under realistic conditions the overall behavior of the proposed routing schemes.

One of the initial objectives is to design a generic routing architecture from where all the specific routing schemes will be derived. The motivations for initiating the architecture work by means of a systematic approach include the following:

- To determine a common baseline with a common architecture that covers all/part of the routing models/schemes.
- To facilitate comparison between the different routing schemes that will be designed.
- To define a common "language"; thus, preventing misinterpretation among different dimensions and actors involved in the design of routing system.
- To lead to modular software development (preventing duplicates).

In other words, by starting from a top-level view down to the design of the routing scheme, it results into a holistic approach of the problem that is complementary to experimental /bottom-up approach. Past experiences show that without a well defined system architecture, adding or removing functionality leads to further complexity (see IP control plane design today); in practice, finding the suitable tradeoff between evolutivity, flexibility, and performance is critical to ensure longevity of the architecture.

A. Definitions

A "*system architecture*" is defined in [4][5][6] as a set of functions, states, and objects/information (referred to as

This research work is conducted and funded by the European Commission through the EULER project (Grant No.258307) part of the Future Internet Research and Experimentation (FIRE) objective of the 7th Framework Programme (FP7). This paper is also partially supported by Spanish MICINN projects FIERRO under contract TEC2010-12250-E and TRION under contract TEC2009-10724.

"elements") together with their behavior, structure (relationships and interactions), composition and spatio-temporal distribution. The specification of these elements is referred to the functional model architecture, the information model architecture and the state model architecture, respectively. Note that the architectural specification includes the principles and guidelines governing their design and evolution over time.

In this paper we focus on the *functional model*. Functional (routing) model determines a systematic decomposition of the (routing) system by defining the routing system functional design, its inputs/outputs, and its various interfaces. This modeling technique (or methodology) enables thus to systematically describe the automated processing that a complex system must perform to transform available inputs to the desired outputs. The fundamental underlying idea is the following: the system is viewed as a distributed computing function. The processing performed by the system can be explained by iteratively decomposing the more complex top-level functions or functional areas into a set of simpler functions (subfunctions). Each subfunction is computed by an organized sub-system. This decomposition is performed up to the level of atomic functions that cannot be further decomposed.

B. *Enhanced Functional Flow Block Diagrams (EFFBD)*

It is necessary to select a technique for generating the different diagrams and decompositions of the functional model. In this case the Functional Flow Block Diagram (FFBD) has been selected and specifically the Enhanced version (EFFBD), which models both the "control" and the "data" aspects of the system [7][8][9].

FFBD provides a hierarchical decomposition of the system's function with a control structure that dictates the order in which the subfunctions can be executed at each level of the decomposition. The FFBD presents thus the logical sequencing of the same subfunctions as those identified through functional decomposition by displaying them in their logical, sequential relationship.

Typically a function shall be represented by a rectangle containing the title of the function (an action verb followed by a noun phrase) and its unique identifier, usually a number. A single arrowhead line indicates the functional flow from one function to another. The AND constructor is a condition in which all preceding or succeeding paths are required. The symbol may contain a single input with multiple outputs or multiple inputs with a single output, but not multiple inputs and outputs combined. See the example in Fig. 1, where F2 and F3 begin in parallel after completion of F1, and F4 begins after completion of F2 and F3.

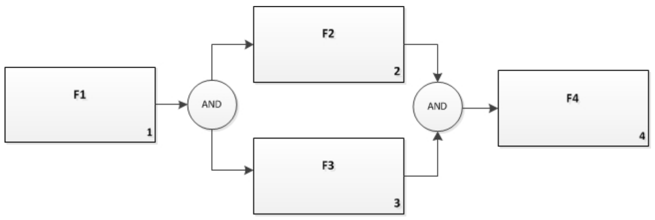

Figure 1. An FFBD example.

In a similar way to the AND constructor there is also an OR constructor which is in fact an exclusive OR: only one of multiple preceding or succeeding paths is required, but not many. An inclusive OR can be build combining AND and OR constructors.

EFFBD displays the control dimension of the functional model in an FFBD format (extended with new constructors: iteration, loops, and multiple exit from functions) with a data flow overlay to effectively capture data dependencies. Thus, EFFBD represents: (1) functions, (2) control flows, and (3) data flows. The logic constructs allow you to indicate the control structure and sequencing relationships of all functions accomplished by the system being analyzed and specified. When displaying the data flow as an overlay on the control flow, the EFFBD graphically distinguishes between triggering and non-triggering data inputs. Triggering data is required before a function can begin execution. Therefore, triggers are actually data items with control implications.

In EFFBD, a function can begin execution if it is both enabled (by control) and triggered (by data). In the case where there is no data trigger specified, a function begins execution upon being enabled. A function is enabled if the function(s) that precedes it in the control flow specification have completed execution (e.g., satisfied their completion criteria). A function is triggered when the required stimulus data item becomes available to the function. See the example in Fig. 2, where F4 is enabled by F1 and triggered by data D2, and F5 is enabled by F2 and "F3 or F4" and triggered by data D3.

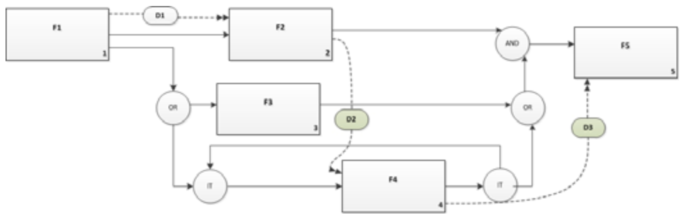

Figure 2. An EFFBD example.

C. *Outline of the paper*

The paper is structured as follows. In Section II we describe our proposed functional model of the routing system architecture. In section III we use this generic architecture to describe a specific scheme of compact multicast routing. Finally, in Section IV we present the conclusions and the future work.

II. DESCRIPTION OF THE ROUTING SYSTEM ARCHITECTURE FUNCTIONAL MODEL

In this section we present our proposal of a functional model of the routing system architecture. The "Route" function is decomposed iteratively in a set of functions and subfunctions (1st and 2nd level decompositions), which are all shown in Fig. 3. Then, for each function, we provide a definition and describe its sequential relationship with other functions using EFFBD diagrams.

The "Route" function is defined as:

- The process of finding a path in a network from any source to any destination along which to send traffic. It

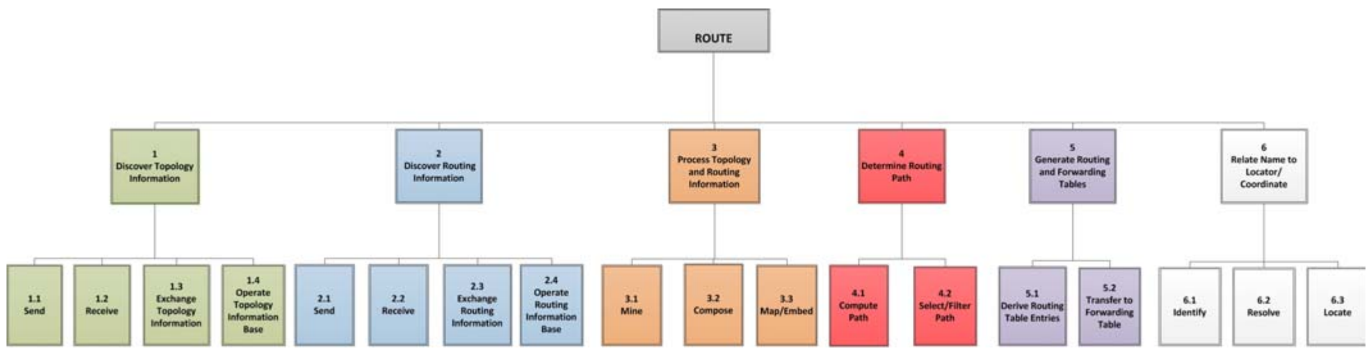
ROUTE
1 Discover Topology Information
2 Discover Routing Information
3 Process Topology and Routing Information
4 Determine Routing Path
5 Generate Routing and Forwarding Tables
6 Relate Name to Locator/ Coordinate
1.1 Send
1.2 Receive
1.3 Exchange Topology Information
1.4 Operate Topology Information Base
2.1 Send
2.2 Receive
2.3 Exchange Routing Information
2.4 Operate Routing Information Base
3.1 Mine
3.2 Compose
3.3 Map/Embed
4.1 Compute Path
4.2 Select/Filter Path
5.1 Derive Routing Table Entries
5.2 Transfer to Forwarding Table
6.1 Identify
6.2 Resolve
6.3 Locate

Figure 3. Functional Hierarchical Decomposition of the Routing System Architecture.

consists in discovering information about the network topology and about the routing paths obtained by other nodes, processing this information, computing new paths and selecting one of them. The results are stored in the Routing Table (RT).

It is decomposed into the following functions (Fig. 3):

- Discover Topology Information (DTI), Discover Routing Information (DRI), Process Topology and Routing Information (PTRI), Determine Routing Path (DRP), Generate Routing and Forwarding Tables (GRFT) and Relate Name to Locator/Coordinate (RNLC).

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 4.

A. *"Discover Topology Information" (DTI) function*

The "Discover Topology Information" function is defined as:

- The acquisition, dissemination and maintenance of information about the topology, i.e., states and properties of interfaces, links and nodes. Topology Information Units (TIUs) are obtained through the exchange of information with other (neighbor and remote) nodes, and from processing this information. The Topology Information Base (TIB) is the database where TIUs are maintained.

It is enabled/triggered by the following functions (Fig. 4):

- Enabled by the DTI function of other nodes, when TIUs are received from other nodes through the external interfaces.
- Auto-enabled by certain changes in the TIB, internal timers or others.
- Enabled by the PTRI function, when its processing results in terms of TIUs are to be stored in the TIB, or when it requests TIUs from the TIB to be processed.

It enables/triggers the following functions (Fig. 4):

- It enables the DTI function of other nodes, when TIUs are sent to other nodes through the external interfaces
- It enables the PTRI function in order to process TIUs for structuring and analysing topology information.
- It enables the DRP function in order to process TIUs for computing new routing paths.

It is decomposed into the following functions (Fig. 3):

- Send and Receive. Sending and reception of TIUs to/from other nodes through the external interfaces, including the operations related to packet scheduling

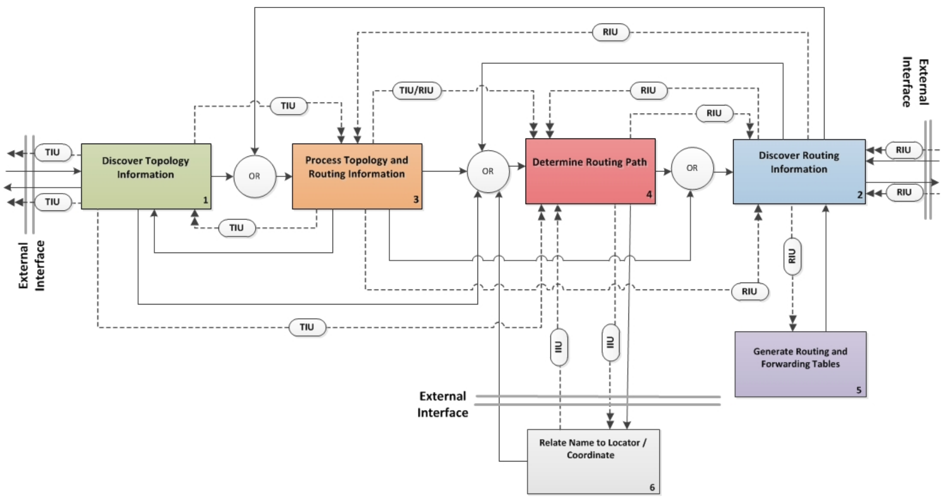
Discover Topology Information
1
OR
Process Topology and Routing Information
3
OR
Determine Routing Path
4
OR
Discover Routing Information
2
Generate Routing and Forwarding Tables
5
Relate Name to Locator / Coordinate
6
TIU
TIU
TIU
TIU
TIU
TIU/RIU
RIU
RIU
RIU
RIU
RIU
RIU
RIU
IIU
IIU
External Interface
External Interface
External Interface

Figure 4. EFFBD of the "Route" function.

and management of input and output queues (storage, sending priorities, discarding rules, etc.)

- Exchange Topology Information. Activation and control of the operations for the acquisition, dissemination and maintenance of TIUs (i.e., it initiates the sending of TIUs to other nodes, the processing of received TIUs and the related operations in the TIB), for the processing of TIUs in order to structure and analyse topology information, and for the processing of TIUs to compute new routing paths. It can be triggered by changes in the TIB, by internal timers or others.
- Operate Topology Information Base. Creation, maintenance and use of the TIB database, including the control to access the data, the enforcement of data integrity, the management of concurrency, the recovery and restoration of the database after failures, and the maintenance of the database security.

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 5.

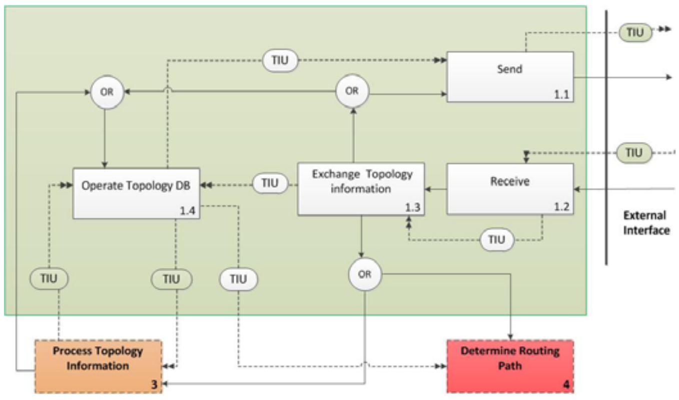

Figure 5. EFFBD of the DTI function.

B. *“Discover Routing Information” (DRI) function*

The “Discover Routing Information” function is defined as:

- The acquisition, dissemination and maintenance of information about the routing paths and/or distances to reachable destinations. Routing Information Units (RIUs) are obtained through the exchange of information with other (neighbour and remote) nodes, and from processing this information. The Routing Information Base (RIB) is the database where RIUs are maintained.

It is enabled/triggered by the following functions (Fig. 4):

- Enabled by the DRI function of other nodes, when RIUs are received from other nodes through the external interfaces.
- Auto-enabled by certain changes in the RIB, internal timers or others.
- Enabled by the PTRI function, when its processing results in terms of RIUs are to be stored in the RIB, or when it requests RIUs from the RIB to be processed.
- Enabled by the DRP function in order to store “computed” and “selected” RIUs in the RIB.
- Enabled by the GRTE function in order to retrieve the “selected” RIUs from the RIB.

It enables/triggers the following functions (Fig. 4):

- It enables the DRI function of other nodes, when RIUs are sent to other nodes through the external interfaces.
- It enables the PTRI function in order to process RIUs for structuring and analysing routing path information.
- It enables the DRP function in order to obtain “selected” RIUs.

It is decomposed into the following functions (Fig. 3):

- Send and Receive. Sending and reception of RIUs to/from other nodes through the external interfaces, including the operations related to packet scheduling and management of input and output queues (storage, sending priorities, discarding rules, etc.).
- Exchange Routing Information. Activation and control of the operations for the acquisition, dissemination and maintenance of RIUs (i.e., it initiates the sending of RIUs to other nodes, the processing of received RIUs and the related operations in the RIB), for the processing of RIUs in order to structure and analyse routing information, and for the selection of RIUs based on some criteria. It can be triggered by changes in the RIB, by internal timers or others.
- Operate Routing Information Base. Creation, maintenance and use of the RIB database, including the control to access the data, the enforcement of data integrity, the management of concurrency, the recovery and restoration of the database after failures, and the maintenance of the database security.

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 6.

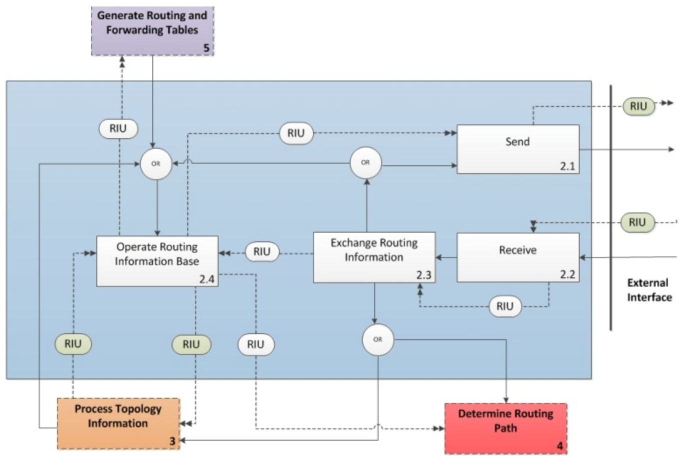
Generate Routing and Forwarding Tables
5
RIU
RIU
RIU
Send
2.1
OR
OR
RIU
Operate Routing Information Base
2.4
RIU
Exchange Routing Information
2.3
Receive
2.2
RIU
External Interface
RIU
RIU
RIU
OR
Process Topology Information
3
Determine Routing Path
4

Figure 6. EFFBD of the DRI function.

C. *"Process Topology and Routing Information" (PTRI) function*

The "Process Topology and Routing Information" function is defined as:

- The structuring and analysis of information about the topology (in terms of TIUs) and the routing paths (in terms of RIUs) using advanced techniques which permit: (1) to obtain patterns, features, properties and hidden relationships in the information, by applying statistical and artificial intelligence methods over large data sets; (2) to derive complex information from simple information; (3) to transform a coordinate space to another space with new properties, which allow to compute better paths

It is enabled/triggered by the following functions (Fig. 4):

- Auto-enabled by means of internal timers and others in order to process information.
- Enabled by the DTI function, when this function requires processing TIUs for structuring and analysing topology information.
- Enabled by the DRI function, when this function requires processing RIUs for structuring and analysing routing path information.

It enables/triggers the following functions (Fig. 4):

- It enables the DTI function for requesting TIUs from the TIB that are to be processed or for storing TIUs in the TIB that have been processed.
- It enables the DRI function for requesting RIUs from the RIB that are to be processed or for storing RIUs in the RIB that have been processed.
- It enables the DRP function in order to process "structured" RIUs or TIUs for computing new routing paths.

It is decomposed into the following functions (Fig. 3):

- Mine. Given a set of TIUs or RIUs, finding of (hidden) relationships, patterns, features/properties or classes between them (in terms of TIUs), by applying statistical and artificial intelligence methods over large data sets.
- Compose. Production of combinations from TIUs or RIUs so as to build more complex TIUs and RIUs (called "structured" TIUs or RIUs).
- Map/Embed. Transformation of a given metric space (in terms of TIUs) into another metric space (in terms of TIUs) with new properties, which allow to compute better paths. The transformation is given by a mapping function.

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 7.

D. *"Determine Routing Path" (DRP) function*

The "Determine Routing Path" function is defined as:

- The computation of new routing paths (called "computed") from the information about the topology (in terms of TIUs) and the routing paths (in terms of RIUs), and the selection of some of these computed paths (called "selected") based on some criteria.

It is enabled/triggered by the following functions (Fig. 4):

- Enabled by the DTI function in order to process TIUs for computing new routing paths.
- Enabled by the DRI function in order to obtain "selected" RIUs.
- Enabled by the PTRI function in order to process "structured" RIUs or TIUs for computing new routing paths.

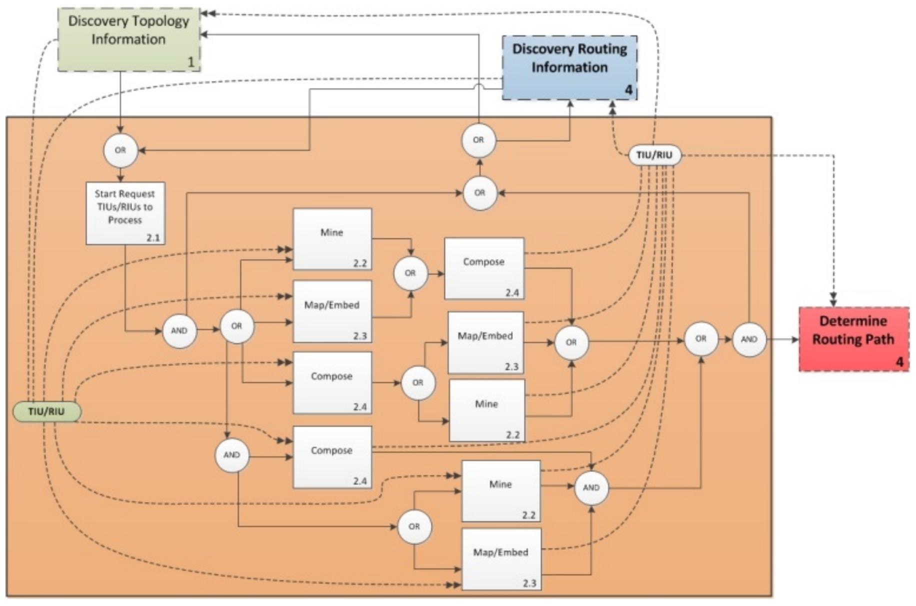

Figure 7. EFFBD of the PTRI function.

It enables/triggers the following functions (Fig. 4):

- It enables the DRI function in order to store "computed" and "selected" RIUs in the RIB.
- It enables the RNLC function in order to resolve or locate a node identifier as well as to identify itself (in terms of IIUs).

It is decomposed into the following functions (Fig. 3):

- Compute. Obtaining of new routing paths (called "computed") in terms of RIUs from ("structured") RIUs and/or TIUs. It can be seen as the operation of finding the routing paths that minimize or maximize a multi-constrained multi-objective function.
- Select/Filter Path. Given a set of RIUs, choice of a limited number of RIUs (called "selected") either by enforcing selection rules, by applying filters, or by multi-criteria decision. Note that RTEs are derived from these "selected" RIUs.

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 8.

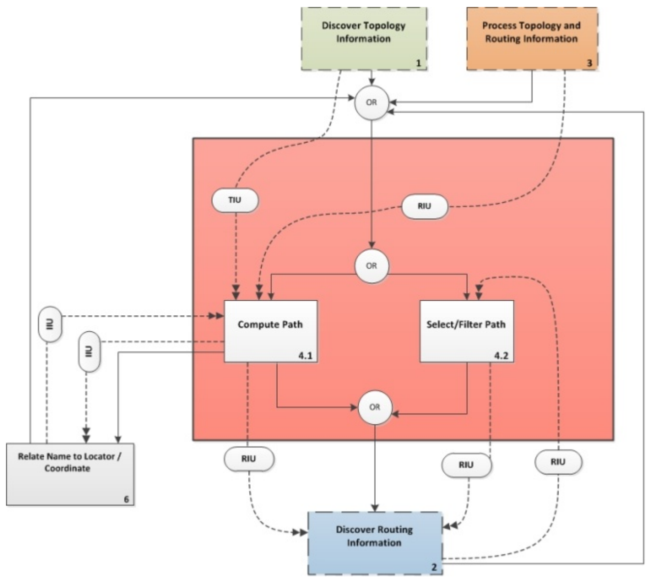
Discover Topology Information
1
Process Topology and Routing Information
3
OR
TIU
RIU
OR
Compute Path
4.1
Select/Filter Path
4.2
IIU
IIU
OR
Relate Name to Locator / Coordinate
6
RIU
RIU
RIU
Discover Routing Information
2

Figure 8. EFFBD of the DRP function.

E. *"Generate Routing and Forwarding Table" (GRFT) function*

The "Generate Routing and Forwarding Table" function is defined as:

- The generation of the Routing Table (RT) and the Forwarding Table (FT). Routing Table Entries (RTEs) are derived from the "selected" routing paths ("selected" RIUs) stored in the RIB, and Forwarding Table Entries (FTEs) are derived from these RTEs.

It enables/triggers the following functions (Fig. 4):

- It enables the DRI function, in order to retrieve the "selected" RIUs from the RIB.

It is decomposed into the following functions (Fig. 3):

- Derive Routing Table Entries. Derivation of the RTEs from the "selected" routing paths ("selected" RIUs) stored in the RIB, and the storage in the RT.
- Transfer to Forwarding Table. Derivation of the FTEs from the RTEs stored in the RIB, and the storage in the FT.

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 9.

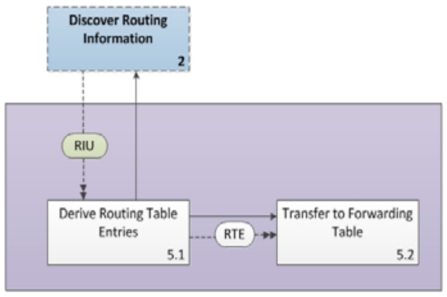
Discover Routing Information
2
RIU
Derive Routing Table Entries
5.1
RTE
Transfer to Forwarding Table
5.2

Figure 9. EFFBD of the GRFT function.

F. *"Relate Name to Locator / Coordinate" (RNLC) function*

The "Relate Name to Locator / Coordinate" function is defined as:

- The identification, resolution and location of nodes (associated to routing), based on Identification Information Units (IIUs). This function is mainly "external" since the identification information is not maintained by the nodes but by an external entity. Internally, in the nodes, this function is responsible of managing the requests of identification, resolution and location with this external entity and/or an internal database where the identification information is cached.

It is enabled/triggered by the following functions (Fig. 4):

- Enabled by the DRP function, in order to resolve or locate a node identifier as well as to identify (in terms of IIUs).

It is decomposed into the following functions (Fig. 3):

- Identify. Assignment of identifiers to nodes. These identifiers can be either topology-dependent (locators) or topology-independent (names). A locator can take the form of a label, a topology-dependent address or a coordinate.
- Resolve. Translation, conversion, or mapping from the name of the destination to its associated locator.
- Locate. The functionality allowing destinations to be located by means of the resolution function.

The sequential relationship between these functions using an EFFBD diagram is shown in Fig. 10.

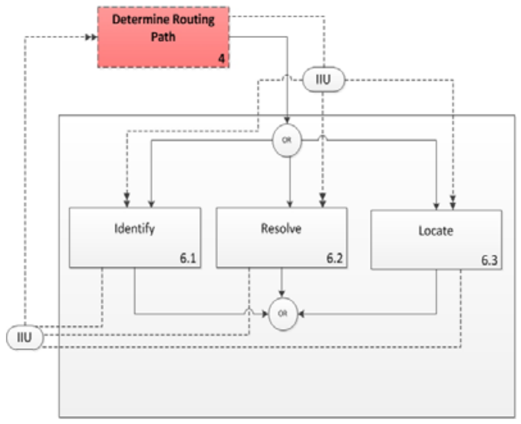

Figure 10. EFFBD of the RNLC function.

III. An Application Example: A Compact Multicast Routing Scheme

A. *Compact multicast routing*

Compact routing schemes address the fundamental tradeoff between the memory space required to store the routing table entries and the length of the routing paths that these schemes produce [10]. As recently formalized in [11], dynamic compact multicast routing algorithms enable the construction of point-to-multipoint routing paths from any source to any set of destinations referred to as leaves. The tree determined by this point-to-multipoint routing path is commonly referred to as the Multicast Distribution Tree (MDT) as it enables the distribution of multicast traffic.

In [12] we recently proposed a name-independent compact multicast routing (CMR) scheme for leaf-initiated, distributed and dynamic construction of MDT. In this context, “leaf-initiated” means that the join/leave requests are initiated by the leaves; “distributed” implies that transit nodes process the join/leave requests and compute the routing table entries (no centralized processing by the root); and “dynamic” refers to the on-line capability to timely process the join/leave requests as they arrive without re-computing and rebuilding the MDT from scratch. The objective of the proposed algorithm is to minimize the routing table sizes of each node $n \in V$ at the expense of i) routing the packets on paths with relative small deviation compared to the optimal stretch obtained by the Steiner Tree algorithm as well as ii) higher communication cost compared to the Shortest Path Tree algorithm. To this end, CMR reduces the local storage of routing information by keeping only (direct) neighbor-related entries rather than tree structures (as in ST) or network graph entries (as in both SPT and ST). As such, it is actually a true “protocol independent” multicast routing scheme. In other terms, the novelty of this algorithm is on maintaining local topology information ($|deg(n)|$ routing table entries) instead of global topology information ($|V-1|$ entries) providing the least cost next hop during the MDT construction. As a first contribution, in [12] we only focused on a leaf node u joining MDT.

B. *Hierarchical decomposition of CMR*

The hierarchical decomposition of CMR is shown in Fig. 11. The problem consists in joining a leaf node u to a multicast tree sourced in s. If node u is already part of MDT then it is either a transit or branching node of the already deployed MDT. Otherwise, node u is not part of MDT and it must search for the least cost branching path towards a node v belongs to MDT.

As said above, nodes do not store topology information and only keeps local storage of routing information knowing only (direct) neighbor-related entries. The information needed to reach a given multicast source s is acquired by means of a discovering search mechanism (details explained in [12]) that returns the upstream node along the least cost branching path to the MDT sourced at s. Such mechanism is triggered whenever a node decides to join a given multicast source s as part of a multicast group.

Two types of messages are involved in this process, namely the request (type-R) messages flowing in the upstream

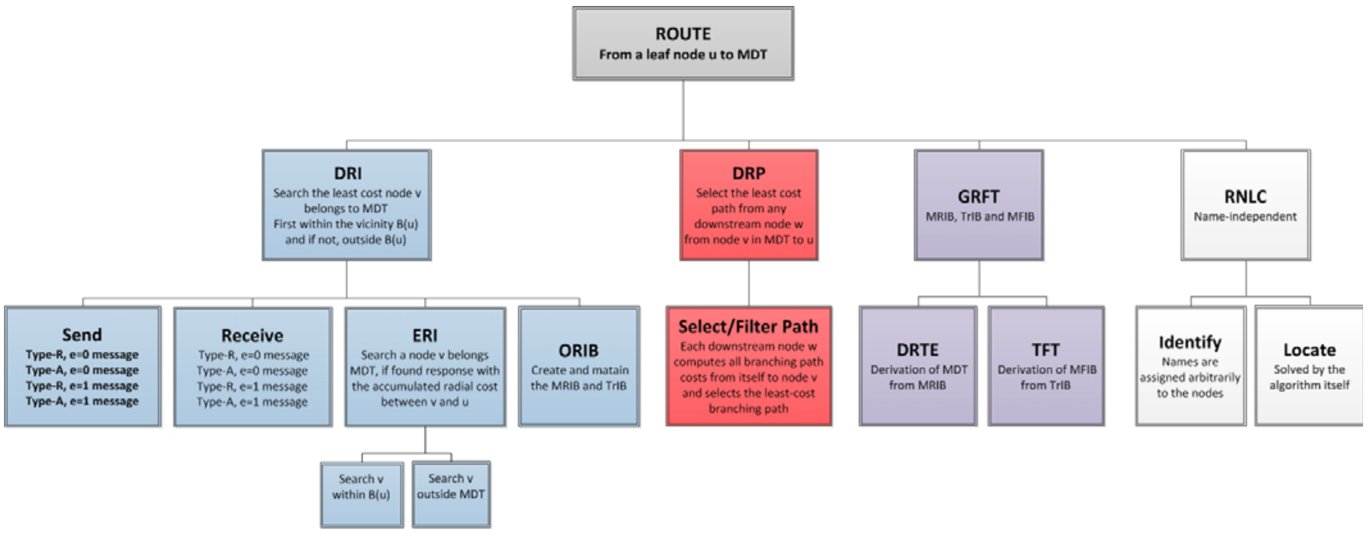
ROUTE
From a leaf node u to MDT
DRI
Search the least cost node v belongs to MDT First within the vicinity B(u) and if not, outside B(u)
DRP
Select the least cost path from any downstream node w from node v in MDT to u
GRFT
MRIB, TrIB and MFIB
RNLC
Name-independent
Send
Type-R, e=0 message
Type-A, e=0 message
Type-R, e=1 message
Type-A, e=1 message
Receive
Type-R, e=0 message
Type-A, e=0 message
Type-R, e=1 message
Type-A, e=1 message
ERI
Search a node v belongs MDT, if found response with the accumulated radial cost between v and u
ORIB
Create and matain the MRIB and TrIB
Select/Filter Path
Each downstream node w computes all branching path costs from itself to node v and selects the least-cost branching path
DRTE
Derivation of MDT from MRIB
TFT
Derivation of MFIB from TrIB
Identify
Names are assigned arbitrarily to the nodes
Locate
Solved by the algorithm itself
Search v within B(u)
Search v outside MDT

Figure 11. Functional Hierarchical Decomposition of CMR.

direction, i.e., towards the multicast source, and response (type-A) messages sent in the downstream direction, i.e., towards the joining leaf node u.

Higher cost may hinder CMR applicability to large-scale topologies such as the Internet. Hence, to keep the communication cost as low as possible, the algorithm's search process is segmented in two different stages by executing first a *local* search covering the leaf's neighborhood (the so-called vicinity B), and, if unsuccessful (no nodes belonging MDT are found), executing a *global* search over the remaining topology. For this very reason, a flag e distinguishes the messages exchanged during the search stages, both type-R and type-A messages are flagged as internal, e=0, if belonging to the local search procedure, and as external, e=1, otherwise. Besides, type-R messages comprise a maximum path budget, pbudget, to define the vicinity B. Being decremented at each hop with the vicinity node's out-degree, pbudget settles the maximum reachability of type-R messages with flag e=0 by determining the size of the vicinity B, whenever pbudget = 0.

This algorithm needs to keep only the Multicast Routing Information Base (MRIB) and Tree Information Base (TrIB) type of routing entries. MRIB is the MDT topology description, i.e., it indicates the upstream neighbor to which to send the join requests along the MDT. After a node becomes member of a MDT, a multicast routing entry is dynamically created and stored in the local Tree Information Base (TrIB). From these routing table entries, multicast forwarding entries are created.

IV. Conclusions and Future Work

In this paper we have presented a proposal of a functional model of the routing system architecture. This generic routing architecture aims to be used as a common framework from which any specific routing scheme can be derived. We have decomposed the route function in a set of subfunctions and described their sequential relationship using EFFBD diagrams. In order to show the validity of the proposal, we have used this generic architecture to describe a specific compact multicast routing scheme.

Next step is the integration of this functional model with an information model (which is already being developed), in order to create a complete architecture of the routing system.

References

[1] EULER research project, EU-FP7-258307, http://www.euler-fire-project.eu.

[2] iLab.t experimental facility at IBBT, http://www.ibbt.be/en/ilab/ilab-t

[3] OFELIA experimental facility, http://www.fp7-ofelia.eu/

[4] D.E.Perry and A.L.Wolf, "Foundations for the Study of Software Architecture", ACM SIGSOFT Software Engineering Notes, Vol.17, No.4, October 1992.

[5] D. Garlan and D. E. Perry, "Introduction to the special issue on software architecture", IEEE Transactions on Software Engineering, 21(4), Apr. 1995, pp. 269-274.

[6] G. Booch, Presentation at the Software Developers' Conference, 1999.

[7] A. McInnes, B. Eames, R. Grover, "Formalizing Functional Flow Block Diagrams Using Process Algebra and Metamodels", IEEE Transactions On Systems, Man, and Cybernetics—Part A: Systems And Humans, Vol. 41, No. 1, January 2011.

[8] J. Long, "Relationships between Common Graphical Representations in System Engineering", 2002 Vitech Corporation, Accessed June 2011, available at: http://www.coinsweb.nl/wiki/ccdownloads/CommonGraphicalRepresentations_2002.pdf.

[9] B.F. Blanchard, BF and W. Fabrycky, "Systems Engineering and Analysis", Prentice Hall, Second Ed., 1990, (Fifth Ed., 2011) and Defense Systems Management College 1990.

[10] D. Peleg and E. Upfall, "A trade-off between space and efficiency for routing tables," J. ACM, vol. 36, no. 3, pp. 510–530, Jul. 1989.

[11] I. Abraham, D. Malkhi, D. Ratajczak, "Compact multicast routing," Proc. 23rd Int. Symp. DISC'09, Elche, Spain, pp.364–378, Sep. 2009.

[12] P. Pedroso, D. Papadimitriou, D. Careglio, "A name-independent compact multicast routing algorithm", available as Technical Report, UPC-DAC-RR-CBA-2011-15, March 2011