FREE - Fast Reroute for Energy Efficiency

Diego Reforgiato, Vincenzo Riccobene
Dipartimento di Ingegneria Elettronica, Elettrica e Informatica – DIIEI
University of Catania, Italy
diegoref@diiei.unict.it, vincenzo.riccobene@dieei.unict.it

***Abstract*— In the last years many efforts in telecommunications network research and design have been devoted to reduce energy consumption of network devices. Development of green routers aimed at saving energy when the input traffic is low: in that sense, a possible approach has been based on putting network components to low power states during idle intervals. In this work we face the problem of further reducing the power consumption in transport networks adopting the Fast Reroute technique together with a low power idle approach, which result in substantial energy saving when the traffic on the network is low.**

***Index Terms*—Fast Reroute, Energy Saving, MPLS, NetFPGA.**

I. INTRODUCTION

Some recent works [1,2] have been proposed in literature to achieve energy efficiency in networks by controlling the usage probability of predefined multiple paths to bring about a situation where some nodes can transition to stand-by state when the traffic volume is small. Moreover, stand-by modes have to be explicitly supported with special techniques to maintain the "network presence" of sleeping components [7]. The Fast Reroute [3] mechanism is a technique, integrated in the MPLS protocol, which detects link failures at the hardware level (without Fast Reroute the OSPF protocol takes several seconds to detect them) and overwrites respective routing table entries with pre-computed routes, to minimize packet drops. In this work we assume that interfaces have different Low Power Idle (LPI) states. Each state corresponds to the maximum bit rate the interface can support and the associated level of power consumption. The idea is to maximize the number of interfaces working in LPI mode (which corresponds to a certain amount of energy saving) in the network routers maintaining the reachability of all potential destinations and maintaining high the network QoS. More in detail, we have extended the Fast Reroute approach in order to detect the hardware interfaces, which can be put to LPI mode (therefore obtaining energy saving). The output traffic of the underlying router can be therefore deviated (depending on the current bit rate and a local control policy) towards other interfaces producing a non-negligible gain of power consumption.

II. THE PROPOSED SOLUTION

In the link state routing protocols, a router gathers information about the entire network topology, and the Djikstra's algorithm is performed using this information in order to calculate the best path for each destination. Using this process the router can detect the best link where forwarding a packet to reach a desired destination. Coupling the Fast Reroute technique, multiple paths to reach each node of the network can be detected. This technique pre-computes alternative routes and includes them in the routing table like backup routes. Therefore, it is possible to individuate two different routes for some destinations: the main route and (if exists) the backup route.

```
function FINDINDINTERF(T)
   for all rows r do
      if sum(r)==1 then
         l ← {j such that T[r,j]=1}
         Ind ← Ind ∪ {l}
      end if
   end for
   for all rows r s.t.  sum(r) > 1 do
      S = {j such that T[r,j]=1 and
           j ∉ Ind}
      if (sum(r) == |S|) then
         l = ChooseInterface(S,T)
         Ind ← Ind ∪ {l}
      end if
   end for
end function
```

Figure 1. Pseudocode of the FindIndInterf function.

In Figure 1 the pseudo-code of the main function ***FindIndInterf*** is shown: such a function takes as input a forwarding table ***T***. In Figure 2(a) an example of ***T*** is provided: basically, each row of the table corresponds to a destination. Each entry of the table contains 1, if the corresponding destination is reachable through that interface, or 0, otherwise. Looking at the Figure 2(a), the reader can notice that each node knows up to two different paths to reach any other destination, information provided by the Fast Reroute algorithm. It may happen, according to the given topology, that only one route towards one or more destinations is found. In such a case (see for example the destination 4 in Figure 2(a)), the considered interface will be indispensable and has to be maintained in the normal operational state (no LPI mode) in order to work with the highest bit rate that the underlying link supports with such destinations.

Destination	int1	int2	int3	int4
1	1	1	0	0
2	0	1	1	0
3	0	1	1	0
4	0	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	0	0
8	0	1	1	0
9	1	1	0	0

(a)

Destination	int1	int2	int4
1	1	1	0
2	1	1	0
3	1	1	0
4	0	0	1
5	1	0	0
6	1	1	0
7	1	1	0
8	1	1	0
9	1	1	0

(b)

Figure 2. Forwarding tables for a generic node.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 257740 (Network of Excellence "TREND") and from the ECONET project (low Energy Consumption NETworks), funded by the EU through the FP7 call.

The very first step of the ***FindIndInterf*** function detects the indispensable interfaces. The first ***for*** statement finds the active interfaces for which the sum of the values of the underlying row is 1. Let us consider the table in Figure 2(a): the destination 4 can be reached only through one interface, ***int4***; this means that ***int4*** is indispensable and will be added to the ***Ind*** set because it cannot stay in LPI mode. After the first ***for*** cycle, it is checked whether the other destinations are reachable through one of the interfaces classified in the previous step as indispensable. If a certain destination cannot be reached by any interface marked as indispensable, then, it will be necessary to mark as indispensable other interfaces, which connect the current node to the underlying destination. When the execution of the algorithm ends, all the interfaces that are not contained within the ***Ind*** set are put to LPI mode. Then, the routing protocol updates all the routing information and a new forwarding table is therefore created. The ***ChooseInterface*** function takes as input the set ***S*** and the forwarding table ***T*** and chooses the interface in ***S*** whose relative column in ***T*** has the highest sum. In our example, after the first execution, the algorithm individuates the ***int3*** interface as non-indispensable, which is therefore put into LPI mode (see Figure 2(b) for the resulting table where the interface ***int3*** has been removed). Of course, the algorithm needs to be run iteratively until it converges (i.e. there are no further interfaces in the node to be added into the ***Ind*** set).

Moreover, each router is equipped with a Network Control Unit (NCU) which checks, for all the non-indispensable interfaces, whether the aggregated traffic cannot sustain the overall incoming bit rate causing packets loss. For such interfaces, if the traffic is above (below) a certain threshold that a given interface in the current LPI mode can support, then the NCU sets the interface to the next (previous) LPI state. The correspondent routing table entries are thus overwritten in order to use the alternative path to reach the destination. These entries will be restored to the initial value by the NCU itself according to the traffic load. Using this approach, the non-indispensable interfaces are used only to receive network packets and the "network presence" is guaranteed.

III. Numerical Results

We run extensive simulations using NS-2 [4] with the primary goal to estimate the average energy gain obtained when putting the interfaces to LPI mode using our technique. We have also simulated the behavior of the NCU. Following this direction, we have initially simulated a random generated topology with a fixed number of nodes: each experiment is characterized by the presence in the network of 20 internal nodes, that form a strongly connected network, and 20 external nodes, each of them connected to one, and only one, internal node. Each external node is, at the same time, the sender of a traffic flow and the receiver of another traffic flow. The obtained topologies have been simulated varying the bit rate of each traffic flow in order to evaluate our approach in different conditions. In terms of power gain, on the average and with the generated topologies, the current results show that about 48% of the interfaces have been marked as non-indispensable (and therefore set to LPI mode) using our technique and, consequently the power consumption has been decreased.

We have compared the energy saving for two different cases: (i) when all the interfaces are set to the default behavior (no LPI mode used) and (ii) when all the non-indispensable interfaces are set to LPI mode. The analysis performed so far indicates that when the current traffic load is lower than the 30% of the entire network capacity, 48% of interfaces can be set to LPI mode, for the given topologies. This scenario is similar to real networks in the nighttime where the application of our technique could save a great amount of energy.

IV. Use Case

In order to provide a real implementation of the proposed approach, we have integrated such a technique in the NetFPGA board, on top of the Fast Reroute and Multipath Routing project [5]. More in detail, we have developed a software module in ANSI C [6], which connects to the process that handles the routing protocol and enables the Fast Reroute feature. Such a program receives all the routing table information and builds a forwarding table similar to the ones previously shown in Figure 2. As in the NetFPGA platform it is not possible to set the LPI mode for any of the four 1G-Ethernet ports, we simply simulated the LPI states with four bit rates: {250Mbps, 500Mbps, 750Mbps, 1Gbps}. Our software periodically issues the commands to set to LPI mode the interfaces that will not be used to send traffic; then the NCU manages the simulated LPI mode of each interface and, if needed, sets them back to the default behavior according to the current bit rate.

References

[1] A. Yamada, S. Imai, and M. Kakemizu. "A path-based traffic control method for green networks". In Proc. of Information and Telecommunication Technologies (APSITT), 2010, 8th Asia-Pacific Symposium, pages 1-5

[2] A.P. Bianzino, L. Chiaraviglio, and M. Mellia. "Grida: A green distributed algorithm for backbone networks". In On-line Conference on Green Communications (GreenCom), 2011 IEEE, pages 113-119, sept. 2011.

[3] P. Pan, G. Swallow, and A. Atlas. "Fast reroute extensions to rsvp-te for lsp tunnels". RFC 4090 (Proposed Standard), Internet Engineering Task Force, May 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4090.txt.

[4] NS-2. http://nsnam.isi.edu/nsnam/index.php/main page.

[5] http://wiki.netfpga.org/foswiki/bin/view/NetFPGA/OneGig/FastRerouteAndMultipathRouter.

[6] http://wiki.netfpga.org/foswiki/bin/view/NetFPGA/OneGig/FREEFastRerouteForEnergyEfficiency.

[7] R.Bolla, R. Bruschi, A. Cianfrani, M. Listanti, "Enabling backbone networks to sleep". IEEE Network, vol. 25, Issue 2, pp. 26-31, 2011.