U.P.B. Sci. Bull., Series C, Vol. 75, Iss. 2, 2013 ISSN 2286 – 3540

ROUTING WITH QoS CONSTRAINTS

Radu MIRUȚĂ[1], Eugen BORCOCI[2]

This paper addresses the problem of routing constraints in overlay networks for both inter and intra domain. It proposes two algorithms (with "blind" style and the shortest path style) for mapping the traffic matrix onto real network graph as to respect the minimum bandwidth constraints and also optimize the resource usage. They were implemented allowing the introduction of numerical values to obtain the number of solved requests, the processing time, the path that satisfies the constraints, indicating the node that produces the bottleneck for the requests remained unsatisfied. Studying by comparison the results for the two proposed algorithms it can be seen the appropriate using situations, parameters that influence the efficiency and limitations of each.

Keywords: constrained routing, resource allocation, Future Internet, content-aware networking, network aware applications

1. Introduction

With over 2.5 billion users worldwide [1] and an increased use by the population of 528% during the period 2000-2011 [2], the current Internet is a great success in terms of connecting people and communities. However, today's Internet was designed in the 1970s for purposes quite unlike today's heterogeneous application needs and user expectations. New layers and technologies continue to be added by extending the scope of what we now consider to be "Internet". Discrepancy between the original design goals and current begins to affect the potential use of the Internet. A great number of challenges in the technology, business, and society must be overcome in developing Future Internet (FI) to support tomorrow's society. Future Internet will enable many new areas of applications that will drive the development of new markets. According to the European Community studies [3] on the FI, the main growth directions of development be considered that the pillars are:

- *Internet by and for people;*
- *Internet of Contents and Knowledge;*
- *Internet of Things;*

[1] PhD student, Electronic Telecommunication and Information Technology Faculty University POLITEHNICA of Bucharest, Romania, e-mail: radu.miruta@elcom.pub.ro

[2] Professor, Electronic Telecommunication and Information Technology Faculty University POLITEHNICA of Bucharest, Romania

- *Internet of Services.*

To support and sustain the growth of these pillars, network infrastructure should be subject to specific research results from a large set of technical challenges associated with network infrastructure. Multimedia services, growing up today dominate the global Internet and there are great sources of income for involved the players. Thus, the need to overcome the inherent weaknesses and limitations of current Internet on providing media services is becoming increasingly imperative. As a response to these needs, the ALICANTE project [4] propose an evolutionary architecture and concept approach oriented towards the development of an Media Ecosystem network representing a flexible medium in which to participate and interaction many service providers, content providers, network providers and end users. The solution enables End-Users to access the offered Media Services in various contexts, within a fully managed environment enabling optimized QoS and QoE, and also to share and deliver their own media content and services dynamically, seamlessly, and transparently to other users [5]. Furthermore, it enables the Network Providers to enrich their offering of network services and to manage more efficiently the network resources, thanks to new Content Awareness capabilities embedded in the network layer. The service-network interaction actually creates a strong cross layer optimisation loop with benefits for both Services and Network actors [5]. According the ALICANTE proposed architecture, more detailed described in [4][5][6], Service Provider initiate a request addressed to the Network Provider containing transport parameters (it expects that a data flow to be transported from node A toward node B being assured a dedicated bandwidth).

In the presented paper we are studying and proposing three algorithms for the mapping and resource reservation problem appeared in this architecture. The problem is the following: *given an inter-domain graph and a Traffic Matrix (TM contains the requests agreed in the SLS), how to map the TM onto real network graph as to respect the minimum bandwidth constraints and also optimize the resource usage.* A similar problem is then solved for each intra-domain case using a similar approach, as Fig. 2 illustrates (VCAN represents a Virtual Content Aware Network established as a logic paths over multiple domains containing traffic with same QoS requirements; CANMgr is the entity which coordinates VCANs in each network domain, more details in [6]):

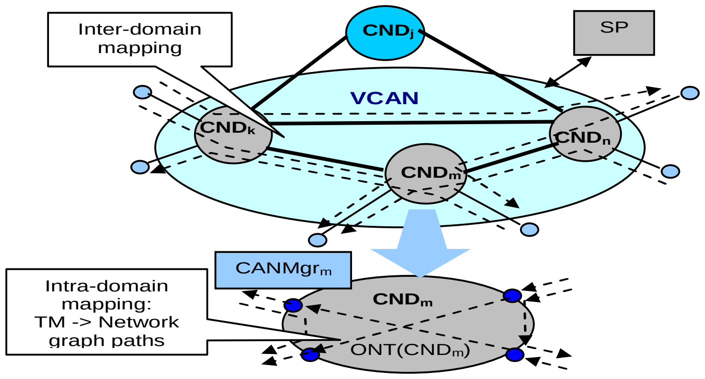


Fig. 1 - Two steps mapping a VCAN

This idea can be considered in the networking context of overlay routing with QoS assuring, or mathematically in the context of multi-commodity flow problem. Overlay topology design has been one of the most challenging research area over the past few years [7]. Most of the efforts on overlay networks can be subsumed under two categories [8]: the first one proposes overlay networks for specific applications in the Internet, and the second category aims at developing generic overlays service networks in the Internet that can be used for a variety of applications. Our efforts belong to the second category where we propose a general overlay service network in [9][10]. In ALICANTE an inter-domain overlay QoS peering and routing [7][10] has been defined. An inter-domain overlay network is first defined, abstracting each domain with a node and its inter-domain links. QoS routing algorithms can then choose QoS capable paths.

2. State of the art

The virtual networks optimal mapping on the resources offered by a physical IP Network is an NP-hard problem [11]. More details about creating, possible benefits and mapping to a Layer 3 network of an overlay network are found in [14]. Our goal is to develop algorithms to map the QoS capable VCANs over several independent network domains in a scalable way to finally assure efficient transport of real-time and media traffic. Protocols are needed to transport QoS and other information between nodes and based on this information, QoS routing algorithms can choose QoS capable paths. Given the ALICANTE project architecture, which is currently using the algorithms presented in this paper, the mapping problem can be divided into two level: inter-domain and intra-domain. In Alicante, an inter-domain overlay QoS peering and routing had been defined [5] [10]. For inter-domain signalling several solutions are proposed (cascade, hub, mixed-mode) [12], [13], but they do not consider the content awareness capabilities of the multiple domain infrastructures. Other solutions for inter-domain QoS peering and routing are based on the overlay network idea [8] [16] [17] [18].

Busted by increasing demand for multimedia applications over Internet, the problem of finding routing paths satisfying QoS constraints has been extensively studied by the research community. Representative examples of such QoS constraints are related to bandwidth, delay, jitter and packet loss [15]. Even if the e-2-e delay is often a metric of interest for multimedia content distribution [15] prefer to minimize delay, maintaining only optimal rate. They proposed an overlay model that represents the real network topology; essentially, it is a regular overlay graph, except the links are not weighted by numbers; instead, the link capacities are variables, and a set of linear capacity constraints express the constraints placed on overlay links by shared bottlenecks. In their model, every node selects d neighbors to which it has links with the highest bandwidth. Similar to our way of establish bandwidth capacity uploaded Intra-NRM to its associated CANMgr for each asked domain, in [15] is used a term of predicted bandwidth. In their work, the highest-bandwidth multicast tree is obtained by a greedy algorithm modified to take linear capacity constraints into consideration.

3. The routing problem

The routing problem appears when it is intended to satisfy the QoS constraints. Receiving a set of requests from the SP side, each one with a bandwidth constraint, it seems to be very important the processing order. Because the path for one request can contain the same segments with the path corresponding to the other request, both of them consuming a part of the available bandwidth, it could be possible that for the last requests from the receiving set do not remain resources. Trying to solve this problem it will be run a constrained routing algorithm. A combined metric is proposed for a link, considering the bandwidth request, the bandwidth available, targeting to choose the widest path. We proposed a cost for the inter-domain link $C(i,j)=\frac{B_{req}}{B_{ij}}=\frac{B_{req}}{B_{available}}$,

where B_{ij} is the available bandwidth for this link and B_{req} is the requested one. Another useful interpretation of this ratio is as link utilization factor; that is the alternative notations will be used: C(i,j) = Ulink_ij. The metric should be ≤1, this representing the bandwidth constraint applied to the algorithm.

3.1. QoS constraints and resources reservation routing algorithm – the "blind" style

Keeping in mind the challenge of solving the resource reservation set of requests for a given network topology, we try in this approach to find the best path of crossing the network graph form the minimum cost point of view solving in the same time as many requests as possible. Considering the graph from Fig. 3, with black lines it is represented the network topology and with red ones, are the requests. Black numbers represent the link capacity and the red numbers are the crossing request used bandwidth.

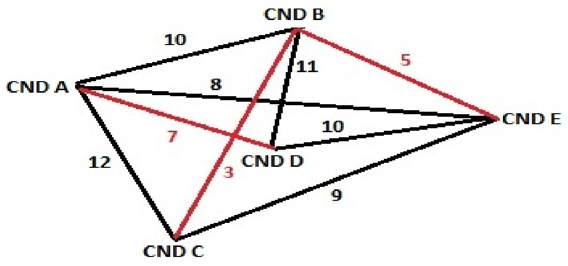


Fig. 2 - Network Topology Graph with a set of requests

Ex : A service provider wants to receive from the above inter-domain network topology a set of reserved paths as follows:

- ✓ from CND **A** – to – CND **D** with a bit rate of 7 Mbps;
- ✓ from CND **B** – to – CND **C** with a bit rate of 3 Mbps;
- ✓ from CND **B** – to – CND **E** with a bit rate of 5 Mbps.

Studying a little bit the network graph and the set of requests also it is easily to observe the arising problem : With which request from the received set it is best to start the reservation process in order to have enough network capacity for serving all requests or as many as possible. In our simple network topology illustrated in Fig. 1, there are 6 different possibilities (3!). For this purpose, we define a metric considering the bandwidth request, the bandwidth available targeting to choose the widest path. The cost of an inter-domain link (i,j) can be C(i,j) = Breq/Bij where Bij is the available bandwidth on this link and Breq is the bandwith requested for that link. The cost of a full path could be Sum (link costs) * number of nodes, where the number of nodes is a weight factor approximately proportional with the number of CNDs crossed by this path.
If we randomly start to serve our requests in 1)CND **A** – CND **D**, 2) CND **B** – CND **C** and 3) CND **B** – CND **E** order, choosing at each step the widest available

path we will be faced with the situation of impossibility to solve the last request because of insufficient bandwidth:

Case A:

$\text{CND A} \overset{7}{-} \text{CND D} : \boxed{\text{CND A} \overset{10}{-} \text{CND B} \overset{11}{-} \text{CND D} : (7/10+7/11)*3 = 4}$

$: \text{CND A} \overset{8}{-} \text{CND E} \overset{10}{-} \text{CND D} : (7/8+7/10)*3 = 4.72$

$: \text{CND A} \overset{12}{-} \text{CND C} \overset{9}{-} \text{CND E} \overset{10}{-} \text{CND D} : (7/12+7/9+7/10)*4 = 8.23$

$\text{CND B} \overset{3}{-} \text{CND C} : \boxed{\text{CND B} \overset{3}{-} \text{CND A} \overset{12}{-} \text{CND C} : (3/3+3/12)*3=3.75}$

$: \text{CND B} \overset{3}{-} \text{CND A} \overset{8}{-} \text{CND E} \overset{9}{-} \text{CND C} : (3/3+3/8+3/9)*4=6.82$

$: \text{CND B} \overset{4}{-} \text{CND D} \overset{10}{-} \text{CND E} \overset{9}{-} \text{CND C} : (3/4+3/10+3/9)*4=5.52$

$: \text{CND B} \overset{4}{-} \text{CND D} \overset{10}{-} \text{CND E} \overset{8}{-} \text{CND A} \overset{12}{-} \text{CND C} : (3/4+3/10+3/8+3/12)*5=8.37$

$\text{CND B} \overset{5}{-} \text{CND E} : \text{CND B} \overset{4}{-} \text{CND D} \overset{10}{-} \text{CND E}$ ✗

$: \text{CND B} \overset{0}{-} \text{CND A} \overset{8}{-} \text{CND E}$ ✗

$: \text{CND B} \overset{0}{-} \text{CND A} \overset{9}{-} \text{CND C} \overset{9}{-} \text{CND E}$ ✗

If we started solving the set of requests in other order, respectively 1) CND **B** – CND **C**, 2) CND **A** – CND **D** and 3) CND **B** – CND **E**, all the requests can be solved:

Case B:

$\text{CND B} \overset{3}{-} \text{CND C} : \boxed{\text{CND B} \overset{10}{-} \text{CND A} \overset{12}{-} \text{CND C} : (3/10+3/12)*3=1.65}$

$: \text{CND B} \overset{10}{-} \text{CND A} \overset{8}{-} \text{CND E} \overset{9}{-} \text{CND C} : \text{cost} > 1.65$

$: \text{CND B} \overset{11}{-} \text{CND D} \overset{10}{-} \text{CND E} \overset{9}{-} \text{CND C} : \text{cost} > 1.65$

$: \text{CND B} \overset{11}{-} \text{CND D} \overset{10}{-} \text{CND E} \overset{8}{-} \text{CND A} \overset{12}{-} \text{CND C} : \text{cost} > 1.65$

$\text{CND A} \overset{7}{-} \text{CND D} : \text{CND A} \overset{7}{-} \text{CND B} \overset{11}{-} \text{CND D} : \text{cost} > 4.72$

$: \boxed{\text{CND A} \overset{8}{-} \text{CND E} \overset{10}{-} \text{CND D} : (7/8+7/10)*3 = 4.72}$

$: \text{CND A} \overset{9}{-} \text{CND C} \overset{9}{-} \text{CND E} \overset{10}{-} \text{CND D} : \text{cost} > 4.72$

$\text{CND B} \overset{5}{-} \text{CND E} : \text{CND B} \overset{11}{-} \text{CND D} \overset{3}{-} \text{CND E}$ ✗

$: \text{CND B} \overset{7}{-} \text{CND A} \overset{1}{-} \text{CND E}$ ✗

$: \boxed{\text{CND B} \overset{7}{-} \text{CND A} \overset{9}{-} \text{CND C} \overset{9}{-} \text{CND E} : (5/7+5/9+5/9)*4=7.28}$

and the total cost for this solution is sum of all costs for each request : 1.65+4.72+7.28 =13.65

***<u>OBS.</u>* Studying more closely, we noticed that the path chosen at each node is more important than the order of solving requests.**

Thus, if in Case A, for CND **A** – CND **D** request we haven't chosen the path CND **A** – CND **B** – CND **D** and we selected the path CND **A** – CND **E** – CND **D** (even if it has a bigger cost), we have solved all the requests even in this case. So the problem appeared when started from point A toward point D, we chosen as next hop point B instead of point E, because its best cost.

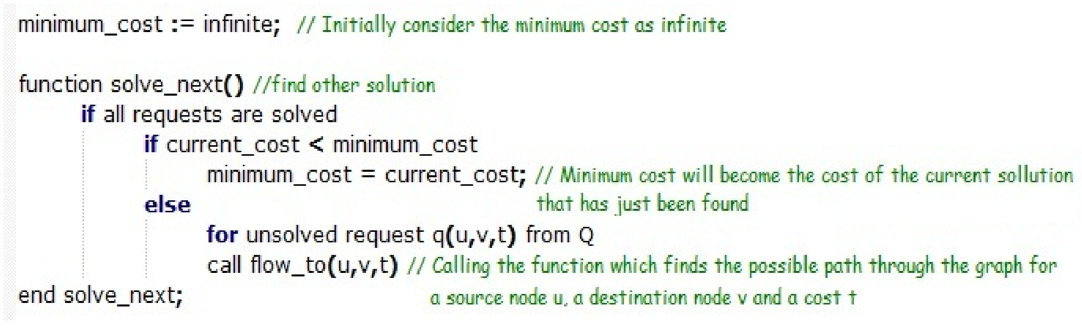
FURTHER, we assume that our goal is to solve as much requests as possible, not only a subset of requests but with the best individually cost.

This approach recursively checks all possible flows for every request, in order to single out the best cost solution targeting the widest available path.

For each request with the source node A and destination B we recursively try to reach node B using depth first search until node B is reached. Using a backtracking approach we find all possible flows from A to B: for each adjacent node with an edge that satisfies the constraints we use a depth first search for the destination node; when this is complete we backtrack to the source node(previous node) of the current node. When the destination is reached we do the same to the next unsolved request and so on. When this operation is complete we continue the backtracking in order to find a different flow. This is done for each request. When a configuration that satisfies all requests is found we compute the total cost.

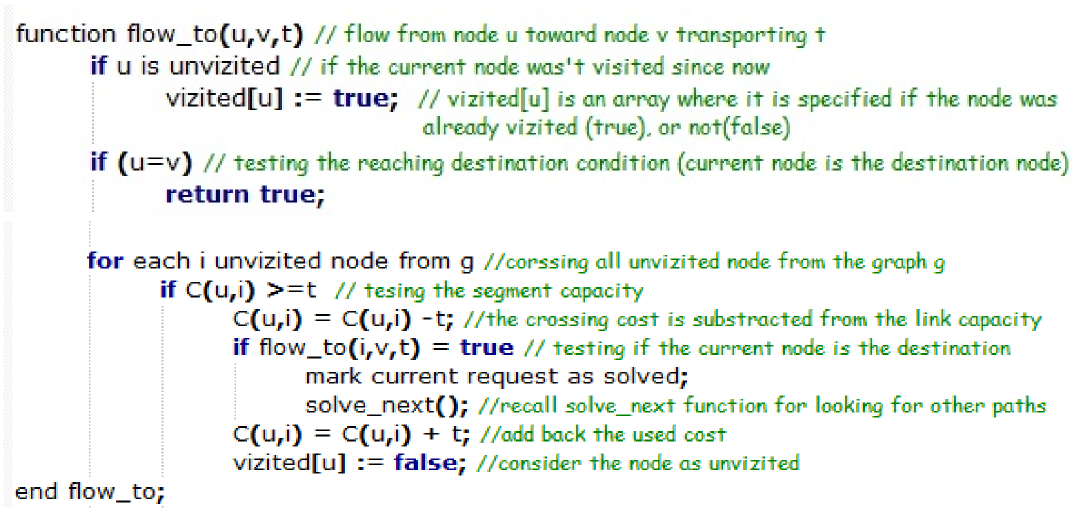
Upon finding a new flow from A to B of a request we compute the partial cost of the solution, using the cost of the requests already satisfied in order to stop using this configuration if the partial cost has already exceeded the best possible cost for a previously found solution.

Pseudo-code:



```
minimum_cost := infinite;  // Initially consider the minimum cost as infinite

function solve_next() //find other solution
      if all requests are solved
            if current_cost < minimum_cost
                  minimum_cost = current_cost; // Minimum cost will become the cost of the current sollution
            else                                  that has just been found
                  for unsolved request q(u,v,t) from Q
                  call flow_to(u,v,t) // Calling the function which finds the possible path through the graph for
end solve_next;                          a source node u, a destination node v and a cost t
```



```
function flow_to(u,v,t) // flow from node u toward node v transporting t
        if u is unvizited // if the current node was't visited since now
                vizited[u] := true;  // vizited[u] is an array where it is specified if the node was
                                        already vizited (true), or not(false)
        if (u=v) // testing the reaching destination condition (current node is the destination node)
                return true;

        for each i unvizited node from g //corssing all unvizited node from the graph g
                if C(u,i) >=t  // tesing the segment capacity
                        C(u,i) = C(u,i) -t; //the crossing cost is substracted from the link capacity
                        if flow_to(i,v,t) = true // testing if the current node is the destination
                                mark current request as solved;
                                solve_next(); //recall solve_next function for looking for other paths
                        C(u,i) = C(u,i) + t; //add back the used cost
                        vizited[u] := false; //consider the node as unvizited
end flow_to;
```

<u>Input dates:</u>

- graph g representing the network topology;
- capacity function C(u,v);
- a set of requests Q(u,v,t); u-current node; v-destination node; t-cost;

Running this algorithm for our graph, from Fig. 3, below can be studied the output (1 means CND A, 2-CND B,..., 5-CND E):

```
==============================
   Fisier de intrare in5.in:
==============================

           Read.
 Solved! Total Cost: 14.962381
          1 3 5 4
           2 1 3
           2 1 5

  Solved! Total cost: 13676488
           1 5 4
           2 1 3
          2 1 3 5

 Solved! Total cost: 18.149242
           1 5 4
          2 4 5 3
          2 1 3 5

Solution No.: 3 Best cost: 13.676588
       Total time: 0.006000
```

Fig. 3 Algorithm output for graph from Fig. 3

As can be observed, there are 3 solutions, the one with the best cost being the second, which is the same that in our example for Case B. The solutions were calculated in 6ms, but the processing time depends on hardware performances. The specified algorithm has been implemented in C, using Visual Studio C++ Express Edition as development environment. The hardware platform used, relevant for the obtained results (especially the average processing time), is a device equipped with Intel(R) Core(TM)2 CPU T5600@1.83GHz processor and 2,00 GB installed memory (RAM) on a 32-bit OS.

The complexity for this algorithm seems to be $c \cdot n^m$, where c represents the no of requests, n is the no of nodes, and m is the no of segments. This algorithm has a major drawback: it computes "in blind" without caring on that it already knows the network graph.

3.2. QoS constraints and resources reservation routing algorithm – the shortest path style

Trying to solve the specified drawback of the previous created algorithm, I further try the development of another algorithm that is based on the shortest path. Because the previous proposed metric is additive, can be modified the Djikstra [19] algorithm to compute the Shortest Path Trees (SPT), one tree for each ingress node where the traffic flows will enter. Note that *Breq/Bij* can be only computed if we know the mapping TT- link (i.e. we know *Breq* for a given link), which is not yet our case. The mapping is to be done jointly with the routing process. So in the first approximation we consider 1/Bij as an additive link metric. Note that other more sophisticated metrics could be considered, e.g. including the delay. The algorithm summary is below (the notation //… represents comments):

```
1. Split the Traffic Matrix TM (requests) in several trees, 1/ingress node ( I1, I2, …In)
2. On the current graph, repeat for 1 to n:
        2.1. Compute the DJ_SPT (root_I1) where DJ means Dijkstra algorithm;
        2.2. Select the TM branches that can be satisfied (i.e. Bij > Breq for that direction);//Mapping and AC
        2.3 Reserve capacities for these branches (subtraction);//a reduced graph is obtained
        2.4. Compute the overall utilization for each path reserved : Upath= Sum_links (Breq/Bavail)
        2.4 List the unsatisfied branches;
        2.5 Try to satisfy the remaining branches by individual depth search ; //additional to DJ algorithm
```

3. Aggregate for all inputs, (satisfied and not satisfied branches) and compute VCAN utilization (sum over all paths mapped onto the real graph);

Optimisation: change order {I1, ..In} and repeat 1..3.

The algorithm starts with splitting the set of requests in subsets characterized by a common source node, i.e. group 1 : 0-3, request 7, group 2: 1-2, request 3, 1-4, request 5 (the nodes A,B,C, .. are denoted as 0, 1, 2, ..). If SP does not specify a wanted order/priorities for request, the algorithm will process them in the received order. An SPT is computed (Dijkstra, using the metric 1/B) for each source node. Then satisfying a request means the find paths on SPT satisfying the requested bandwidth. After each solving of a request the graph is updated by subtracting the reserved bandwidth. SP will be finally informed about the requests non-satisfied and this is a matter of further negotiations. The steps are repeated for all other subsets as described in the summary of the algorithm. Normally the NP would like that for a given traffic matrix associated to a VCAN, the best mapping would be that one having the least overall utilization. Therefore a straightforward optimization method is to recomputed the is to compute the step 2 of the algorithm for other order of inputs given by the bijective function f(I1, ..In) -> {Ik1, Ik2, ..Ikn} which creates actually permutations of the set {I1, ..In}. The function is random. The selection of the best allocation is done by taking the mapping solution having the least overall utilization. Note that for large n, the solution is not scalable (we would need n!) computations. Therefore in practical cases one can stop repetitions of the step 2 after some computations if the overall utilization fulfil some enough good threshold fixed by local policy.

```
                 ==============================
                   Fisier de intrare in26.in:
                 ==============================
                        Input file read.

                    Request 0->3, cost 7: 0 4 3
                    Request 1->2, cost 3: 1 0 2
    Request 1->4, cost 5 unsatisfied on 0->4, avail.cap.1 path
                          traveled: 0 4

              Cost: 6.375000 Satisfied requests 2 / 3
           -------------------------------------------
                    Request 1->2, cost3: 1 0 2
                    Request 1->4, cost 5: 1 0 4
   Request 0->3, cost 7 unsatisfied on 0->4, avail.cap. 3. path
                         traveled: 0 4 3

             Cost: 5.667857 Satisfied requests: 2 / 3
```

```
-------------------------------------------

            Best cost: 5.667857
         Satisfied Requests: 2 / 3
            Total time: 0.003000
```

Fig. 4 Algorithm output for graph from Fig. 3

The complexity for this algorithm is $c! \cdot n^2$, where c represents the no of group of requests and n is the no of nodes.

4. Comparative results

Because of the complexity of this problem, the algorithm results depend a lot of different versions of input dates like number of nodes of the topology, number of links between these nodes, number of requests, desired requests value versus network load. The performance also depends on number of subsets with the same source node, number of requests in the same subset and so on. A lot of simulations have been performed with different set of parameters. *However note that this algorithm does not have to run in real time given that it is used at provisioning actions.* To analyse the performance several input files have been generated containing different size of graphs with different loads, different size of requests with different requirements.

Table 1

Results for the blind style algorithm

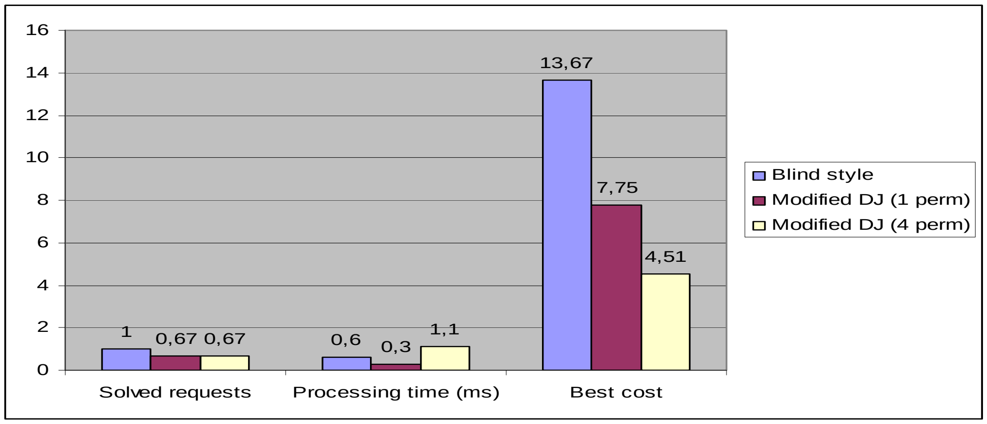
Blind style algorithm	Solved requests	Time(s)	No solutions
5 nodes; 3 requests	3/3	0,006	3
50 nodes; 4 requests	4/4	0,518	48
75 nodes; 4 requests	4/4	Undetermined	Undetermined

Table 2

Results for modified Dj applied for various topologies (4 permutations)

Modified Djikstra style	Max. no of solved requests	Processing time(s)	Permutation no
5 nodes; 3 requests	0,67	0,010	4
50 nodes; 4 requests	1,00	0,021	4
75 nodes; 4 requests	0,75	0,027	4
75 nodes; 7 requests	0,71	0,058	4

Chart 1 - Blind vs. modified Djikstra style (5 nodes and 3 requests)



Studying these results we can observe that the blind style algorithm solve 100% requests for the same topology where the second algorithm only for 67%. For only one order of solving requests, the modified Djikstra algorithm solve 2/3 requests in an aproximative half time comparing with the blind one. We can also observe that even if it is the same topology, the best cost varies from one processing order of requests to other one. In both of these orders the number of solved requests is the same (0.67 <- > 2/3, but at a different costs. More that that, even if the number of them is the same, the solved requests are different. In both cases the last request can not be honored, but this last request differ from one case to another. As we expected, for the same network topologies, the time is increasing for a bigger number of permutations.

5. Conclusions

The presented sections propose each one algorithm for mapping the TM onto real network graph as to respect the minimum bandwidth constraints and also to optimize the resource usage. From the performance point of view of each algorithm we noted the parameters that influence the efficiency, the appropriate use situation, and also the limitations of each one. The one named by us “blind” always solves all requests if it exists this possibility while the one which is looking for the shortest path doesn’t investigate if there could exist other way of solving the unsatisfied requests. Instead, this one can find solutions regardless the graph size while we noted that the first one doesn’t succeed for some sizes. Numerical examples obtained from preliminary implementations of the algorithm are given, showing the variability of performance with the graph complexity, number of requests and order of evaluation. Given the NP-hard characteristics of the problem, several effort towards heuristic solutions improvement should still be done, while considering the network policies. The solution is currently under complete design, evaluation and implementation inside the FP7 ALICANTE project.

REFERENCES

[1] http://www.future-internet.eu/home/future-internet-assembly.html
[2] http://www.internetworldstats.com/stats.htm (last accessed 27.03.2012)
[3]http://www.nessieurope.eu/files/PositionPapers/ETP%20Vision%20on%20Future%20Internet.pdf (last accessed 27.03.2012)
[4] http://www.ict-alicante.eu
[5] *Anne-Lore MEVEL* (eds.) "Overall System ând Components Definition ând Specifications", ICT ALICANTE, Deliverable D2.1, 2010.
[6] *E.Borcoci, M.Stanciu, D.Niculescu, D.Negru, G. Xilouris*, "Connectivity Services Management in Multidomain Content-Aware Networks for Multimedia Applications " , Proc. of. INTERNET 2011, Luxembourg, June 2011.
[7] *D. Adami, C. Callegari, S. Giordano, G. Nenocioni, M. Pagano*, "Design Performance Evaluation of Service Overlay Network Topologies", Journal of Networks, Vol. 6, No.4, April 2011
[8] *Z. Li, P. Mohapatra*, Qron: QoS-aware routing in overlay networks", Selected areas in COmmunications, IEEE Journal on, vol. 22, no. 1, pp. 29-40, 2004
[9] *E. Borcoci, R. Miruta, S. Obreja*, "Network Resources Allocation in Content-Aware Networks for Multimedia Applications, Intetnet 2012, Venice, Italy
[10] *E. Borcoci, R. Miruta, S. Obreja*, "Inter-domain Peering in Content-Aware Networks for Multimedia Applications", CTRQ 2012, Chamonix, France
[11] *A. Haider, Richard Potter and A. Nakao,* "Challenges in Resource Allocation in Network Virtualization", 20th ITC Specialist Seminar, 18.-20. May 2009, Hoi An, Vietnam, http://www.itcspecialistseminar.com/paper/itcss09Haider.pdf.
[12] *P. Levis, M. Boucadair, P. Morrand, and P. Trimitzios*, "The Meta-QoS-Class Concept: a Step Towards Global QoS Interdomain Services", Proc. of IEEE SoftCOM, Oct. 2004.
[13] *M. Boucadair, et al*., "A Framework for End-to-End Service Differentiation: Network Planes and Parallel Internets", IEEE Communications Magazine, Sept. 2007, pp. 134-143.
[14] *I. Stoica*, http://www.cs.virginia.edu/~cs757/slidespdf/757-09-overlay.pdf
[15] *Y. Zhu, B. Li*, "Overlay Networks with Linear Capacity Constraints", University of Ontario
[16] *Fabio L. Verdi, Maurıcio F. Magalhaes*, "Using Virtualization to Provide Interdomain QoS-enabled Routing", Journal of Networks, April 2007, pp. 23-32.
[17] *Z. Li, P. Mohapatra, and C. Chuah*, Virtual Multi-Homing: On the Feasibility of Combining Overlay Routing with BGP Routing, University of California at DavisTechnical Report: CSE-2005-2, 2005.

[18] ENTHRONE, End-to-End QoS through Integrated Management of Content, Networks and Terminals, FP6 project, URL: http://www.ist-enthrone.org/.
[19] http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm