Future Generation Computer Systems 25 (2009) 142–146

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A path finding implementation for multi-layer networks

Freek Dijkstra [a,*], Jeroen van der Ham [a,b], Paola Grosso [a], Cees de Laat [a]

[a] System and Network Engineering Group, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

[b] TNO Defence, Security and Safety, Postbus 96864, 2509 JG Den Haag, The Netherlands

ARTICLE INFO

Article history:
Received 5 February 2008
Received in revised form
8 July 2008
Accepted 12 July 2008
Available online 23 July 2008

Keywords:
Path finding
Multi-layer networks
Network modelling

ABSTRACT

The goal of the OptIPuter project is to tightly couple research applications with dynamically allocated paths. Since OptIPuter is a multi-disciplinary project, the paths through the network often span multiple network domains, and the applications are challenged to find valid network connections through these domains.

The challenge arises if the different network domains use different technologies. In this case, we have a multi-layer path finding problem.

We will show that there are situations where algorithms as used in single layer networks, such as BGP, SS7 and OSPF-TE, cannot find the shortest path. A shortest path in a multi-layer network can contain loops, and a segment of a shortest path may not be a shortest path in itself.

To solve this problem, both a multi-layer network representation as well as new path finding algorithms need to be developed. An additional challenge is to make a generic path finding algorithm that is technology-independent, and does not need to be modified as new technologies emerge.

We show that it is possible to create solutions for all three problems. Using RDF-based techniques, we model multi-layer networks and describe incompatibilities for the path finding algorithm in technology-independent way. We also present a path finding algorithm that is able to use this information to find valid paths.

1. Problem statement

Fig. 1 shows an example multi-layer network. Each circle in the picture represents an administrative domain. The domains are interconnected by links: the edges in the figure.

This example is based on a historic scenario, although the topology is modified to emphasise our point.

The example network contains two types of links: Gigabit Ethernet (GE) and OC-192 links. OC-192 connections are based on SONET technology and carry 192 STS channels. Each STS channels can carry roughly 51 Mbit/s, so multiple STS are needed to carry 1 Gbit/s Ethernet. The Ethernet coming from the two universities is transported over the OC-192 links by embedding (adapting) Ethernet in STS channels. Unfortunately, multiple standards exists that describe how to do this adaptation. In our example, CA*net can embed Gigabit Ethernet in 24 concatenated STS channels (STS-24c adaptation), while NetherLight can embed Gigabit Ethernet in 21 STS channels (7 virtual containers of 3 concatenated channels, STS-3c-7v). StarLight supports both methods to adapt Ethernet in STS channels.

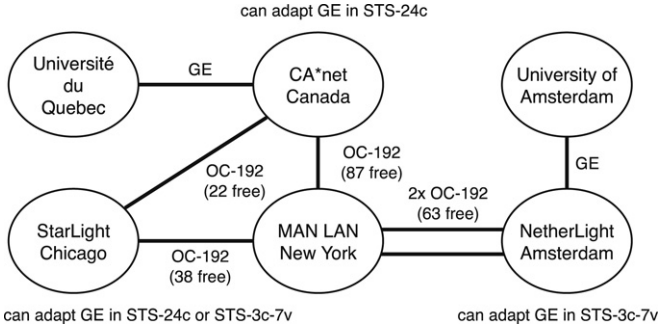

Fig. 1. Example of a multi-layer and multi-domain network, extended from an example presented in [1].

We claim that the shortest path from the Universitédu Quebec to the University of Amsterdam contains a loop — it uses the same physical link twice.

The path Quebec – CA*net – MAN LAN – NetherLight – Amsterdam is not valid since the adaptation performed at CA*net, adaptation of Ethernet in 24 STS channels, is incompatible with the adaptation of Ethernet in 21 STS channels, as performed in NetherLight. We observe that this incompatibility occurs between

* Corresponding address: Advanced Internet Research, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands. Tel.: +31 20 5257531.
E-mail address: fdijkstr@science.uva.nl (F. Dijkstra).

0167-739X/$ – see front matter © 2008 Dr Freek Dijkstra. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.07.002

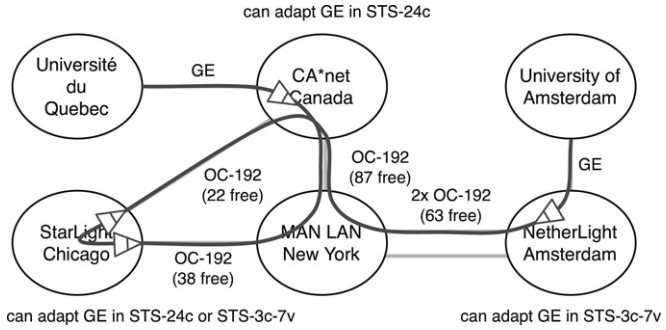

Fig. 2. The shortest valid network connection from Université du Quebec to University of Amsterdam through the example network of Fig. 1.

two domains (CA*net and NetherLight) that are not directly connected to each other.

Furthermore, the path Quebec – CA*net – StarLight – MAN LAN – NetherLight – Amsterdam (with conversion between STS-24c and STS-3c-7v at StarLight) is not valid because CA*net adapts Ethernet in 24 STS channels, and only 22 STS are available between CA*net and StarLight.

The shortest valid path is Quebec – CA*net – MAN LAN – StarLight – CA*net – MAN LAN – NetherLight – Amsterdam, as shown in Fig. 2. This shortest path contains a loop: the link CA*net – MAN LAN is used twice. In the section CA*net – MAN LAN – StarLight, Ethernet is adapted in 24 STS channels, while the section StarLight – CA*net – MAN LAN – NetherLight embeds Ethernet in 21 STS channels. Also, observe that this second section is not a shortest path in itself. The shortest path between StarLight and NetherLight, is StarLight – MAN LAN – NetherLight, even when incompatible adaptations and bandwidth restrictions are taken into account. Thus a segment of a shortest path does not have to be a shortest path in multi-layer networks.

The reason for the strange behaviour as we have seen (loops and the fact that a shortest path does not need to be composed of other shortest paths) is that multi-layer path finding is a *path-constrained* problem. Single layer algorithms such as path vector algorithms in SS7 [2] and BGP [3] or link state algorithms in OSPF-TE [4] can only deal with *link-constrained* problems. In a link constrained problem, the possibility to use each edge is independent from the use of other edges. In a path constrained problem, the possible use of an edge depends on the choice of other edges in the path. For example, data can only be extracted (de-adapted) from a lower layer if there is a corresponding adaptation function earlier in the path.

2. Multi-layer network description language

A simple graph only describing devices as nodes and links as edges is not sufficient to describe multi-layer networks. The basic problem is that multi-layer networks have at least three fundamental 'building blocks' (devices, links and adaptations), while graphs only provide two 'building blocks' (vertices and edges). In earlier work, we defined a multi-layer network model [1] based on ITU-T Recommendation G.805 functional elements [5] and the label concept in GMPLS [6]. This combination of functional elements and labels is also present in the recent ITU-T Recommendation G.800 [7]. Our main contribution is that we can describe the state of a network (such as G.800 and G.805 can), and also describe how this state can be changed: the capability of the network, as required for path finding.

We turned this abstract network model in a syntax [8] by extending the existing Network Description Language, NDL [9,10].

Rather than defining technology-specific ontologies, we first described generic properties of multi-layer networks based on our abstract model. This ontology contains the classes Layer, Adaptation (including multiplexing and inverse multiplexing), and Labels.

Since the technology descriptions refer to generic concepts, a path finding algorithm only needs to know those generic concepts and does not need to be adjusted as new technologies emerge.

Using the network ontology and technology description, network administrators can describe their networks. Each administrator can publish information about their own network, either with full details or only an abstracted view of their network. Because our syntax is based on RDF [13], the domains can link to each others network descriptions using the RDF `seeAlso` property.

3. Technology descriptions

The multi-layer network description has been published earlier [10], but we did not have experience with the actual usage at that time. The effectiveness of our model depends on the ability to model existing protocols in this simple network. Most technologies could easily be described in the model, despite the limited number of classes (only Layer, Adaptation and Label). We created technology descriptions for technologies WDM, SONET, SDH, ATM, fibre and Ethernet. OTN is missing from this list, since we did not have enough experience with those standards.

The descriptions of these technologies proved possible in nearly all situations, and we defined some guidelines how to model two incompatible encodings of a certain technology.

(1) If two encodings can never appear in conjunction on the same link, model it as two different layers. For example, no SONET interface can automatically switch between OC-48 and OC-192, so we modelled OC-48 and OC-192 as two distinct layers.
(2) If the compatibility appears in different labels, model it as a label. For example the difference between tagged and untagged Ethernet is the presence or absence of a label.
(3) If the incompatibility has many distinct options (such as a MTU from 1518 to 16 114 bytes), then we model it as a layer property to avoid an explosion of possible adaptations or possible layers.
(4) If two encodings occurs only in combination with one or a few other layers, we model it as a different adaptation. For example, we model the different spacings of WDM systems (CWDM, DWDM with 25, 50 or 100 GHz spacing) as distinct adaptations.
(5) If all alternatives are exhausted, model it as a (technology-specific) layer property.

Different packet sizes in Ethernet was the only incompatibility we could not capture using our model. We can describe it in NDL, but as a technology-specific layer property. This means that the technology independent path finding algorithm can not take it into account, and has to be handled at the signalling phase.

While describing Ethernet and WDM technologies, we encountered two non-straightforward encodings. Tagged Ethernet is described as Ethernet in Ethernet. Additional logic was required to describe interfaces which can accept both untagged, tagged and Q-in-Q Ethernet [11] at the same time (such as the Ethernet interface in CA*net, which only adapts it in STS channels but does not process the individual frames). The additional logic defines when a label is absent (untagged Ethernet). The calculation of available wavelengths in WDM was non-trivial, since the DWDM standard [12] defines an infinite number of wavelengths, with non-constant spacing between the different wavelengths. The most simple solution is to force interfaces to list the set of all available wavelengths.

Furthermore, we defined the switching capability of devices using only two parameters: the layer and label-conversion capability. For example, most Ethernet and WDM switches can not convert between VLAN tags or wavelengths, while most STS

devices are able to convert between timeslots. Ethernet required the definition of a third parameter, to distinguish between unicast, multicast and broadcast switches.

The generic concepts even allowed us to model "technologies" such as fibre ducts (a bundle a fibres through the same trench in the street), effectively providing information for shared risk link groups.

4. Path finding

There are at least two path finding algorithms through multi-layer networks, a variant of the breadth first search algorithm and a variant of a k-shortest path algorithm [14]. The first algorithm operates on a graph which is similar to the abstract network description we used (and has $\mathcal{O}(|N| \times |Y|)$ vertices, with $|N|$ the number of nodes, and $|Y|$ the number of network layers). The second algorithm operates on a graph which which has significantly more vertices, one for each possible technology stack for each node ($|V| \approx \mathcal{O}(|N| \times T^{|Y|})$ with T the number of incompatibilities per layer). A network domain in the first graph can be mapped to vertices and edges using only local knowledge of the domain. A domain in the second graph can only be described with topology knowledge of all other domains.

We have chosen to implemented the first breadth first search algorithm, because its mapping from NDL description to graph was more straightforward, and could be done using only local knowledge of a domain. We feed this algorithm a representation of the example network of Fig. 1 and tested if it could find the shortest valid path.

Fig. 3 shows a graphic representation of our multi-layer network model for our example network. In this figure, the circles are logical interface and the diamonds switch matrices. The black squares are domains or devices. The layers are represented by different colours. Physical interfaces are mapped to an adaptation stack with multiple logical interfaces.

The result of the path find algorithm is shown in Fig. 3 as well. The coloured path is the shortest path, traversing Quebec – CA*net – MAN LAN – StarLight – CA*net – MAN LAN – NetherLight – Amsterdam. Observe that the link CA*net – MAN LAN is indeed traversed twice, as represented by a thicker line.

The algorithm we use is basically a breadth first search algorithm. It starts at interface `if1-eth` at Quebec, and the list of possible path is expanded with one hop this is repeated till `if1c-vc4` at CA*net where the possible paths branches at the switch matrix. Each branch is tried at the same time, hop by hop. There are two possible locations where a path can branch: at switch matrices and at interfaces that support multiple adaptations, such at `if2-vc4` in StarLight, which can adapt to either `if2-eth7` and `if2-eth8`. Branches are terminated if they can never lead to a viable shortest path. Terminations occur if (a) the layers do not match; (b) at de-adaptations: if the adaptations do not match; (c) at a switch matrix without label conversion: if the labels are not compatible; and (d) at (inverse) multiplexing adaptations: if no more channels are available.

5. Algorithm variants

Path finding is part of a four-step process to set up network connections. The steps are (1) routing, the distribution of the state of a device or domain to its neighbours; (2) path finding, determine (a) viable path(s) using the given information; (3) select a path and determine its parameters that have not been decided upon; (4) path provisioning, configuring the actual network elements. NDL provides a way to distribute routing information. This article mostly deals with step 2, path finding. Steps 3 and 4 are not covered at this time.

Table 1
Number of iterations required by different algorithm variants to find the shortest path in network of Fig. 1

Algorithm variant	Number of iterations
Unrestricted flooding	$1.6 \times 10^{15} \pm 36\%$ iterations (estimate)
Flooding without direct loopbacks	$18 \times 10^{12} \pm 27\%$ iterations (estimate)
Explicit direction (ingress/egress)	587 iterations
No repeated stacks in path	486 iterations
Shortest path to stack only	245 iterations
Interfaces used once per path	No result found (False negative)
Adaptation restriction only (ignore labels)	False positive after 219 iterations
Topology restriction only	False positive after 134 iterations

If our algorithm is not terminated when the shortest path is found, other branches will continue to try new paths. This may yield other possible paths, and the algorithm turns into a k-shortest path finding algorithm.

The path finding mechanism described in this article provides an exact result. Its focus is on accuracy as opposed to speed. It is possible to remove some of the branch termination logic. For example, by not counting the available channels or not checking for compatible labels. This may give a few false positives: paths that contain incompatibilities and can not be used in practice. This may not be a problem if it is combined with a k-shortest path variant, since alternative paths may yield a correct solution. Alternatively, it is possible to increase the number of branch termination rules. While this reduces the flooding nature of our algorithm, it may result in false negatives (no path found, while one is in fact available). For example, it is possible to terminate if a node is processed twice. This assumes that a shortest path never contains loops, even though that is possible as we have seen in our example network. It is also possible to terminate if a node is processed twice *with the same adaptations stack*. This assumes that segment of a shortest path is also a shortest path.

Changing the branch termination rules may increase the number of false positives and false negatives, but reduce the time complexity of the algorithm. For example, the last two rules in previous paragraph would limit the number of branches in the algorithm to $\mathcal{O}(|N| \times |Y|)$ and $\mathcal{O}(|N| \times |S|)$ respectively, with $|N|$ the number of nodes (domains, devices or interfaces), $|Y|$ the number of layers, and $|S| \approx T^{|Y|}$ the number of possible technology stacks. Without these branch termination rules, the number of branches may grow exponentially without upper limit.

Table 1 lists a few results for protocol variants. The first four variants return an exact result, with various degree of suppression of the broadcast mechanism. The middle variants have even greater suppression, but not find the correct path. The last two variants combine this suppression with loosened restrictions, which may yield false positives.

GMPLS has a clear decoupling between routing (OSPF-TE [4]), path finding (PCE [15]) and signalling (RSVP-TE [16]), and works with limited information (OSPF-TE does not distribute information about available labels). The stitching framework developed in Gèant2 [17] is also geared towards the signalling phase, where the choice for parameters are made. The stitching framework defines rules about matching or non-matching parameters and describing technology-specific parameters in terms of these common rules. This is a very similar approach as we used in NDL, were technologies are described using a common model. Both NDL and the stitching framework are technology independent.

The flexibility of our approach can be seen by another variant of path finding algorithm, the path walk algorithm. This variant does not traverse the *possible connections*, but the *currently configured connections*. Effectively, it finds the current configured

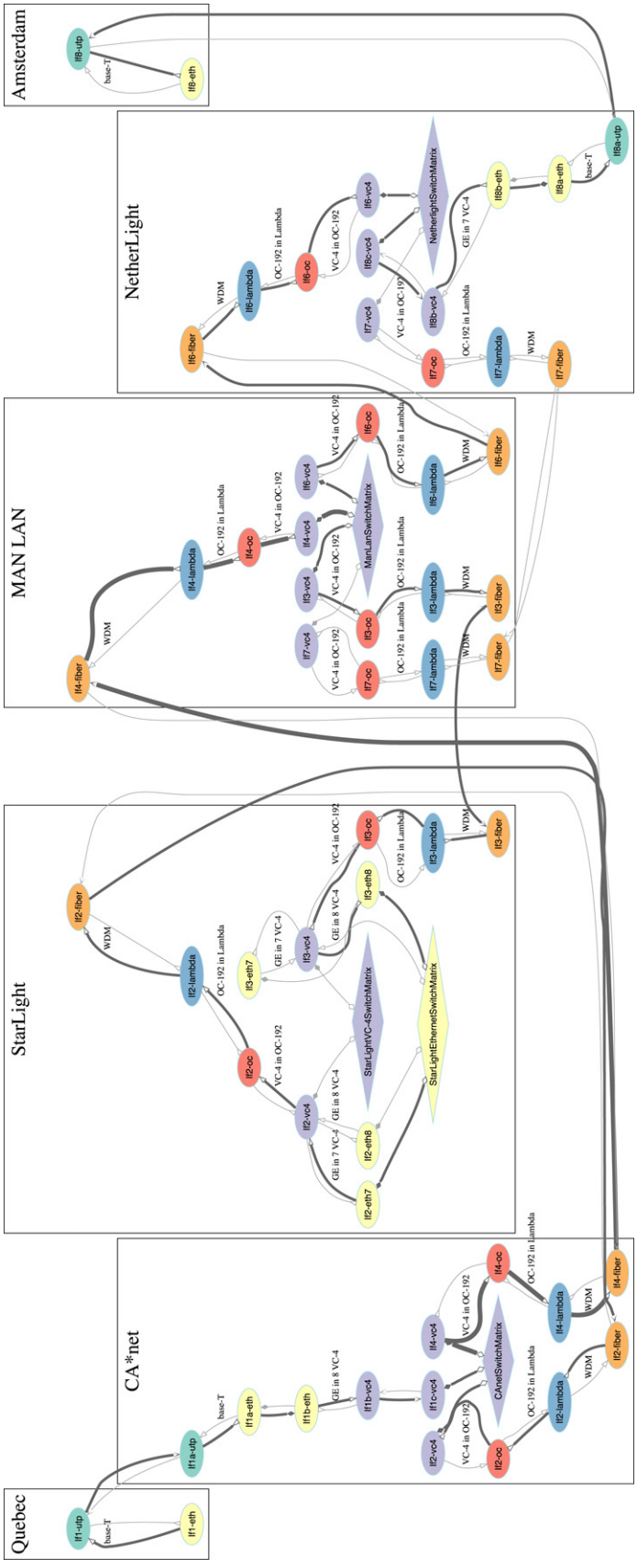

Fig. 3. The visual output of an automated path finding algorithm.

network connections instead of the potential available network connections. We use this path walk algorithm as the basis for a fault isolation framework which is able to detect anomalies in the published network configuration, and thus detect and isolate faults across domains.

6. Conclusion

We have shown a working implementation of a breadth first search algorithm that can find paths in multi-layer networks. Among the lessons we have learned during its development are:

- A shortest path in a multilayer network can contain a loop (the same link is used twice)
- A segment of a shortest path in a multilayer network does not have to be a shortest path by itself.
- Multi-layer path finding is a path-constrained problem. Single layer path finding is a link-constrained problem.
- A simple graph only describing devices as nodes and links as edges is not sufficient to describe multi-layer networks.
- If the network description uses a technology independent model, the path finding algorithm does not need to be adjusted as new technologies emerge.
- SONET, SDH, ATM, Fibre, WDM, and the VLAN part of Ethernet technologies can be described using only three base classes: Layer, Adaptation and Label.
- Network descriptions based on RDF can be distributed, as domains can link to each others using the RDF `seeAlso` property.
- Our RDF-based network description can not only describe the state of the network, but also how that state can be changed. This is required for path finding.
- The capability description only required three parameters for a switch matrix: the layer, label conversion capability and – for Ethernet – distinction between unicast, multicast and broadcast.

Our implementation can even if the shortest path contains a loop or if a segment of a shortest path is not a shortest path in itself. Using this algorithm, applications in the OptIPuter project will be able to find dedicated network connections between different domains, even if those domains use different technologies.

Acknowledgments

Part of this research is done under the GigaPort Next Generation project led by the Dutch National Research Network (SURFnet), and the Interactive Collaborative Information Systems (ICIS) project. Both projects are supported by the Dutch Ministry of Economic Affairs (grant numbers BSIK03020 and BSIK03024).

The authors wish to thank the anonymous reviewers for their helpful suggestions. Bert Andree and Karst Koymans have been very helpful in establishing the abstract multi-layer model. Niels Roosen aided in programming parts of the path walk algorithm.

References

[1] F. Dijkstra, B. Andree, K. Koymans, J. van der Ham, C. de Laat, A multi-layer network model based on ITU-T G.805, Computer Networks 52 (10) (2008) 1927–1937. doi:10.1016/j.comnet.2008.02.013.
[2] Signalling network functions and messages, Recommendation ITU-T Q.704, International Telecommunication Union (ITU), July 1996. URL: http://www.itu.int/rec/T-REC-Q.704/.
[3] Y. Rekhter, T. Li, S. Hares, A border gateway protocol 4 (BGP-4), RFC 4271 (Draft Standard), Jan. 2006. URL: http://www.ietf.org/rfc/rfc4271.txt.
[4] K. Kompella, Y. Rekhter, OSPF extensions in support of generalized multi-protocol label switching (GMPLS), RFC 4203 (Proposed Standard), Oct. 2005. URL: http://www.ietf.org/rfc/rfc4203.txt.
[5] Generic functional architecture of transport networks, Recommendation ITU-T G.805, International Telecommunication Union (ITU), March 2000. URL: http://www.itu.int/rec/T-REC-G.805/.
[6] E. Mannie, Generalized multi-protocol label switching (GMPLS) architecture, RFC 3945 (Proposed Standard), Oct. 2004. URL: http://www.ietf.org/rfc/rfc3945.txt.
[7] Unified functional architecture of transport networks, Recommendation ITU-T G.800, International Telecommunication Union (ITU), September 2007. URL: http://www.itu.int/rec/T-REC-G.800/.
[8] J. van der Ham, F. Dijkstra, Network description language homepage. URL: http://www.science.uva.nl/research/sne/ndl/.
[9] J. van der Ham, F. Dijkstra, F. Travostino, H.M. Andree, C. de Laat, Using RDF to describe networks, Future Generation Computer Systems 22 (8) (2006) 862–867. doi:10.1016/j.future.2006.03.022.
[10] J. van der Ham, F. Dijkstra, P. Grosso, R. van der Pol, A. Toonk, C. de Laat, A distributed topology information system for optical networks based on the semantic web, Journal of Optical Switching and Networking 5 (2–3) (2008) 85–95. doi:10.1016/j.osn.2008.01.006.
[11] Provider bridges, IEEE Standard 802.1ad, IEEE, May 2006. URL: http://www.ieee802.org/1/pages/802.1ad.html.
[12] Spectral grids for WDM applications: DWDM frequency grid, Recommendation ITU-T G.694.1, International Telecommunication Union (ITU), June 2002. URL: http://www.itu.int/rec/T-REC-G.694.1/.
[13] Resource description framework (RDF). URL: http://www.w3.org/RDF/.
[14] F. Kuipers, F. Dijkstra, Path selection in multi-layer networks, Computer Communications, 2008 (in press), URL: http://staff.science.uva.nl/~fdijkstr/publications/multilayer-pathselection.pdf.
[15] A. Farrel, J.-P. Vasseur, J. Ash, A Path Computation Element (PCE)-based architecture, RFC 4655 (Informational), Aug. 2006. URL: http://www.ietf.org/rfc/rfc4655.txt.
[16] L. Berger, Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions, RFC 3473 (Proposed Standard), Jan. 2003. URL: http://www.ietf.org/rfc/rfc3473.txt.
[17] A. Escolano, A. Mackarel, D. Regvart, V. Reijs, G. Roberts, H. Popovski, Report on testing of technology stitching, Deliverable DJ3.5.3, GÉANT, May 2007. URL: http://www.geant2.net/upload/pdf/GN2-07-066v5-DJ3-5-3-Report_on_Testing_of_Technology_Stitching.pdf.

Freek Dijkstra received his M.Sc. in applied physics from the Utrecht University in 2002. He is researcher in System and Network Engineering group at the University of Amsterdam, where he pursues a Ph.D. degree. Freek's primary interest is path finding through multi-layer optical networks.

Jeroen van der Ham received his M.Sc. degree in cognitive artificial intelligence from Utrecht University in 2002, and the M.Sc. degree in system and network engineering from the University of Amsterdam in 2004. In 2004, he joined the System and Network Engineering group at the University of Amsterdam, where he is currently pursuing a Ph.D. degree. His primary research interests are network topology representations, the network control plane, and its multi-domain aspects.

Paola Grosso received her Ph.D. in Physics from the University of Turin — Italy. She worked at the Stanford Linear Accelerator Center and since 2004 she is a member of the SNE (System and Network Engineering) group of the University of Amsterdam. She is leading the group effort in the fields of optical networking and her current research interests are lightpath provisioning and resource scheduling, and semantic network models. Dr. Grosso is chair of the OGF NML-WG, Network Markup Language Working Group.

Cees de Laat is associate professor and group leader of the System and Network Engineering Science group in the Informatics Institute at the University of Amsterdam. Research in his group includes optical/switched networking for Internet transport of massive amounts of data in TeraScale eScience applications, RDF to describe networks and associated resources, distributed cross organization Authorization architectures and Systems Security. With SURFnet he develops and implements projects in the GigaPort Research on Networks. He collaborates in the NSF - OptIPuter project. He served in the Open Grid Forum as IETF Liaison and co-chair of the Grid High Performance Networking Research Group (GHPN-RG). He is co-founder and organizer of several of the past meetings of the Global Lambda Integrated Facility (GLIF) and founding member of CineGrid.org.