This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

A Service-Transparent and Self-Organized Optical Network Architecture

Kyriakos Vlachos and Apostolos Siokis

Computer Engineering and Informatics Department &
Research Academic Computer Technology Institute, University of Patras, Rio, Greece
(Tel: +30 2610 996990, Fax: +30 2610 969 007, email: kvlachos@ceid.upatras.gr)

Abstract— **In this paper, a new service oriented networking paradigm is presented, where network nodes (peers) are self-organized into individual service entities. The key idea relies on the overlay approach, where there exists a virtual service plane, fragmented into self-organized and self-managed entities called islands of service transparency. The islands are formed in an upstream, ad-hoc mode from the non-networking resources (i.e VoD, grid server, etc) towards all ingress routers of the network, using link state advertisements and multi-cost path selection algorithms (i.e residual bandwidth, server capacity, storage, etc). Organization and re-organization of nodes around non-network resources is transparent to end-users, and thus any request within a specific service island is transparently routed to the island's resource for execution. A service proxy is commissioned to resolve service addresses and service attributes to QoS metrics. In this paper, we present the main notations and metrics of the proposed architecture as well as node behavior and potential GMPLS extensions for implementation issues.**

***Index Terms*— service transparency, optical transparent networks, path selection, self-organization**

I. INTRODUCTION

The term *self-organization* occurs in many branches of science, yet there is no commonly accepted definition in the literature. Researchers study existing phenomena of self-organization, but they take different viewpoints and analyze different aspects [1]. In the general case, *self organized* networks are defined as a cluster of peers located in any (OSI) layer that are capable of cooperation in order to achieve a higher goal. In physical layer cooperation functions may be defined as some measures taken for characterizing link capacities, node capabilities, and the link status. In Data Link Control layer cooperation may be introduced by using some physical resources allocation scheme, providing better facilities with respect for security (secure control channel etc). In network layer cooperation may involve any aspect of the data transmission as researched by many authors. A network is *self-organized* if it is organized without any external or central dedicated control entity. In other words, the individual nodes interact directly with each other in a *distributed* peer-to-peer fashion (*localization*). Another important aspect of self-organized networks is their ability to adapt to changes as for example to link or node failures, [2] or even to attacks [3]. In fact, the nodes continuously adapt to changes in a coordinated manner, such that the system always reorganizes as a reaction to different internal and external triggers for change.

Albeit self-organization in communication network and especially in wireless sensor networks have been fairly enough researched, [4] such a concept has not been used before in optical networks apart from attack avoidance [3]. This is because optical networks are considered to be static with point-to-point connections. Even when dynamic lightpath establishment is employed to increase efficiency, still the transport layer remains service un-aware in the sense that there is no interplay between the so called service plane and data/control plane, [5]. To this end, features like joint resource optimization with resiliency, [6], or complete service restoration instead of simple path restoration, are difficult to be implemented and require cross layer optimization [7]. In addition, a service may consist of different attributes that can be of different nature and it is unclear how to convert them to QoS criteria of both network (i.e bandwidth, delay, etc) and non-network (i.e. VoD capacity, liability, CPU cycles, etc) resources based on SLAs signed between the users and the service providers.

In this paper, we propose a new networking paradigm, termed as SO-SOON ("*Self-Organizing, Service Oriented Optical Network*"), to introduce service awareness in the core optical network, by creating *self-organized islands of service transparency*. A *service island* consists of a single (non-network) resource and a group of networking nodes that constitute the shortest path towards that resource. This group of nodes is service transparent and thus upon a service request, end-users' data are transparently forwarded, to the island's resource and not outside it. The proposed architecture and particularly the *service islands* are self managed entities in the sense that core nodes are self-organized in an ad-hoc fashion, based on multi-criteria path selection algorithms, thus adapting themselves to updated networking or non-networking conditions, [8]. In this paper, we analyze the proposed network paradigm and define basic notations, metrics as well as we analyze how the networking peers (nodes) interact to each other to form self-managed ad-hoc entities. We also provide potential path selection algorithms that determine island organization also discuss implementation issues and potential GMPLS extensions, [9].

The work described in this paper was carried out with the support of the BONE-project ("Building the Future Optical Network in Europe"), a Network of Excellence and the DICONET-project, funded by the European Commission through the 7th ICT-Framework Program.

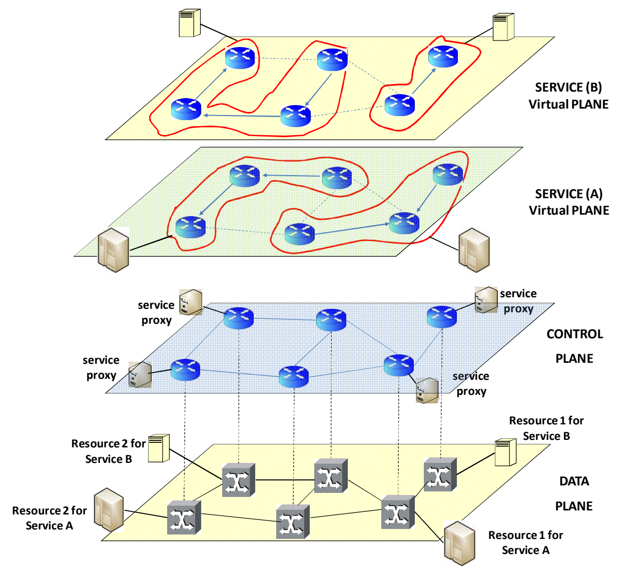


Figure 1: SO-SOON network paradigm employing service transparent optical islands.

II. Optical Islands of Service Transparency

Consider the abstract concept of a network. Its formal definition is *G(V,E)*, where *G* denotes the network graph, *V* is its set of vertices and *E* is its set of edges. This definition is extended to *G(V,E,S)* with the addition of the overlaying services, running on top of the network infrastructure. We denote as $S = (S_1, S_2, \dots, S_N)$ that set of services. The provision of each service to end-user means that there exist service specific resources in the network that can be computing elements, storage, VoD servers etc. Each resource can be replicated more than one time in the network either for survivability issues or for serving more or different customers. We denote the k^{th} resource for i service as S_i^k and relying on the overlay approach, there exist a virtual service plane for each individual service in the network. Each service plane is fragmented into self-organized and self-managed entities hereinafter called *service transparent islands*. Figure 1 displays graphically the SO-SOON approach. Two kind of services are offered, denoted as service A and B. Each one possesses two set of resources denoted as resource #1 and resource #2, connected to specific egress nodes. For each service, there exists a virtual service plane (replica of the control plane), where nodes are self-organized per resource in such a way that each service request within that network domain is transparently routed to that resource for execution. A *service proxy* is responsible for *service addressing* and is placed at the entry points of the network.

In the original network graph *G(V,E)*, the self organizing schema is identified by ordering in a efficient way the vertices in order to create subgraphs where their biconnected set is empty. In other words, there is a unique path for each source-destination. To this end, in the proposed approach, the use of self-managed islands per service indicates the construction of multiple partitions (or subgraphs of *G*) based on a distributed function or algorithm (see Figure 2). The islands are formed in

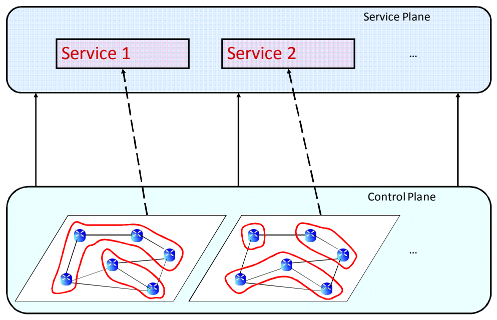


Figure 2: Construction of multiple sub-graphs over a network topology G(V,E) per service.

upstream mode from the resource towards an ingress router (or any router that has no upstream node), using link state advertisements that include information from both the networking elements (i.e residual bandwidth from the nodes) and the resource elements (i.e VoD server capacity, storage, etc). Each core node behaves as an edge node against the island of nodes and autonomous decide if it will join or not an island. In fact, the node receives state information from different downstream nodes and based on some performance criteria, decides which island to join. It then forwards the decision and the update cost performance metrics upstream. With the addition of a node, the island is extended and the new node is considered part of the relevant service plane.

In the proposed scheme, one needs to exchange information between the virtual service planes and the actual data/control plane. This is necessary to resolve services and resources to specific destinations hosts. In addition, a service may consist of different *service attributes* that can be of different nature, including for example both network and/or non-network values. These attributes are service specifics and accompany the service requests submitted by the end users. They can represent the Service Level Agreements (SLA) between the service provider and the end-user. To this end, the service proxy role is first to resolve the user request to identify the type of service and thus determine the destination host. Secondly, to resolve the SLAs of that request to provide relevant cost metrics and thus determine if the request can be accepted or not. In that manner, the service proxy can automatically provide QoS by design and not by policy in its vicinity.

Service discovery is another key aspect of the whole design since end-users should not interfere or show preference, which resources are to be used. The service islands are self-organized and re-organized to direct requests to any resources, depending on the performance metrics used when created. For example, when a video server is overloaded with a high number of subscribed users, then it can be the case, nodes to re-organize themselves to point to another, less congested video server. The prime idea is that users should not experience the difference and will be totally un-aware of where their requests are routed. The network control that the service proxy may exercise can be easily integrated into the edge router functionality by extended its capabilities. In general, the service proxy actions include:

- On the network level: resolve service and service specific attributes to provide a destination address and QoS metrics to the network.
- On the application level: it communicates via an API with the service plane(s) to register the available services. In fact, the service proxy communicates with the rest service proxies in a distributed, non centralized manner to discover services, thus creating a service routing table.

It is obvious that the service routing graph is a subset of the network graph but it may contain more detailed information with respect to a service. The basic idea behind service proxy is service routing according to resource availability. If for instance, more than one resource of the same service exists in the network, the service proxy should direct the client based on the decision not only which path is better but also which server has less load, better capabilities etc. This information is communicated to the service proxy upon the formation of the service island and thus service proxy becomes aware of the address of the destination resource as well as of other networking parameters (i.e. bandwidth, delay, number of hops, etc.)

III. NOTATIONS AND SELF-ORGANIZATION OF NODES

As mentioned before, a network is modeled as a graph *G(V,E,S)*. A node can be either a host or a router. The destination hosts are resources that provide the requested services.

Definition. *Local Network State Information.*

For each network node R_k, k=1,2,..V, there exists a local network state information represented as $D[R_k]$, relevant to the performance of the on-demand service $S_r \in$ S, r=1,2,..N. The local network state information $\overline{D}$ generally assigns several components (metrics) related to the network element R_k and is hereinafter called *distance cost*. For example, for a certain service type, e.g. S_1, a resource can be 'allowed' or 'prohibited', being an example of a Boolean routing policy. Generally, however, $\overline{D}$ will be a vector of values. The candidate components of the vector $\overline{D}$ can be divided into four groups: those that accumulate and those that don't; those that change with traffic conditions and those invariant of traffic load. The *distance vector cost*, field denotes local network state information $\overline{D}$ that contains the values of the parameters taken into account to reach a destination address. These parameters are service specifics and constitute attributes to be requested, when the user request is submitted. These attributes can be categorized in network (such as bandwidth (B), end-to-end delay (D) and hop-count (h)) and non-network specific ones (such as CPU cycles or storage capacity for grid services).

Problem definition: *Service aware routing problem formulation in self organized optical networks.* Suppose that a user located at the ingress node R_i requests service $S_r \in S$, with the specific attributes (service specifics) represented by the following constraint vector:

$$J = \left[B_{job}, h_{job}, D_{job}, CPU_{job}, ST_{job}\right]^T$$

with entries being the *bandwidth*, *hop count*, *delay*, *CPU cycles* and *storage capacity*. Assuming that $S_r = (S_r^1, S_r^2, \ldots, S_r^K)$ is the set of resources for service *r*, the problem is to find a path (or paths) $P(R_i, S_r^k)$ with $k \in [1, \ldots, K], S_r \in S$, $S_r \in S$, such that:

$$\overline{D}\left(P(R_i, S_r^k)\right) \leq J \Rightarrow \begin{bmatrix} B_{R_i \to S_r^k} \\ h_{R_i \to S_r^k} \\ D_{R_i \to S_r^k} \\ CPU_{R_i \to S_r^k} \\ ST_{R_i \to S_r^k} \end{bmatrix} \leq \begin{bmatrix} B_{job} \\ h_{job} \\ D_{job} \\ CPU_{job} \\ ST_{job} \end{bmatrix} \qquad \text{Eq. 1}$$

For comparing multi-dimensional metrics put on paths, we refer to the notion of lattices [8]. The comparison with $\leq$ between the two vectors of path information follows that: $\left(B_{R_i \to S_r^k} \geq B_{job}\right) \cap \left(h_{R_i \to S_r^k} \leq h_{job}\right) \cap \left(D_{R_i \to S_r^k} \leq D_{job}\right) \cap \left(CPU_{R_i \to S_r^k} \geq CPU_{job}\right) \cap \left(ST_{R_i \to S_r^k} \geq ST_{job}\right)$. In order to create end-to-end path vectors, we rely on using additive, multiplicative or limited vector operators. For example hops and delay are cumulative costs in the sense that going from node R_k to R_{k+1}, the relevant cost of the path vector is $P_{R_1,R_2} = \begin{bmatrix} D_{R_1} \\ h_{R_1} \end{bmatrix} + \begin{bmatrix} D_{R_2} \\ h_{R_2} \end{bmatrix} = \begin{bmatrix} D_{R_1} + D_{R_2} \\ h_{R_1} + h_{R_2} \end{bmatrix}$, while for limiting parameters like storage capacity or bandwidth, it is: $P_{R_1,R_2} = min[R_1, R_2]$.. Note that although *hop count* is a parameter that has to do with the number of hops traversed, it could be seen as parameter associated to links by assigning $h[R_1, R_2] = 1$ for every link $[R_1, R_2]$.

Having all the above in mind, the network could be seen as two graphs: one graph containing the non-network resources (computing elements, storage area, VoD servers etc), and the other one the network resources (edge and core network equipment ie routers, switches, etc). The edges of the graphs are inter-connected with a unique *score function* F_s (discussed in the next section). This score function is calculated using the aforementioned parameters that constitute the distance vector cost. Thus, every router is linked with every resource with a score. If a router does not lead to a destination, then the score towards that resource is zero. For example, in Figure 3, node R_1 points to only a single resource denoted as S_r^1 with a score F_{S_1}. Similarly node R_2 points to two resources (S_r^3 and S_r^n) with different scores but finally points to only one (S_r^3).

Each node stores its distance cost vector and communicates it, upstream to be updated until it reaches an ingress node. Figure 4 graphically shows the end-to-end shortest paths/tunnels formed starting from the egress nodes of Figure 3. The distance vector cost for node R_i that is adjacent to egress node R_k is defined as follows:

$$\left\{\overline{D}\left(P(R_i, S_r^k)\right) = \overline{D}\left(P(R_i, R_k)\right) \circ \overline{D}\left(P(R_k, S_r^k)\right)\right\}$$

$$= \begin{bmatrix} B_{R_i \to R_k} \\ h_{R_i \to R_k} \\ D_{R_i \to R_k} \\ - \\ - \end{bmatrix} \circ \begin{bmatrix} B_{R_k \to S_r^k} \\ h_{R_k \to S_r^k} \\ D_{R_k \to S_r^k} \\ CPU_{S_r^k} \\ ST_{S_r^k} \end{bmatrix} = \begin{bmatrix} min\left\{B_{R_i \to R_k}, B_{R_k \to S_r^k}\right\} \\ h_{R_i \to R_k} + 1 \\ D_{R_i \to R_k} + D_{R_k \to S_r^k} \\ CPU_{S_r^k} \\ ST_{S_r^k} \end{bmatrix}$$

However, node R_i does not necessarily point to resource S_r^k. Because of its self-managed attitude it may point to any resource of service S_r, through any neighbor node (R_k') that maximizes score function, F_s, as follows:

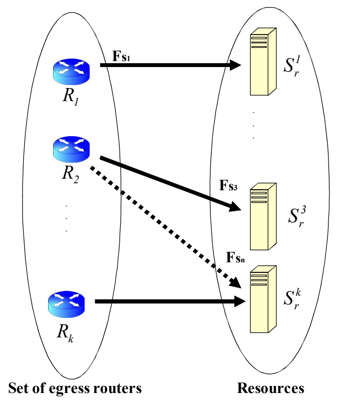


Figure 3: Representation of the network as two sets of node

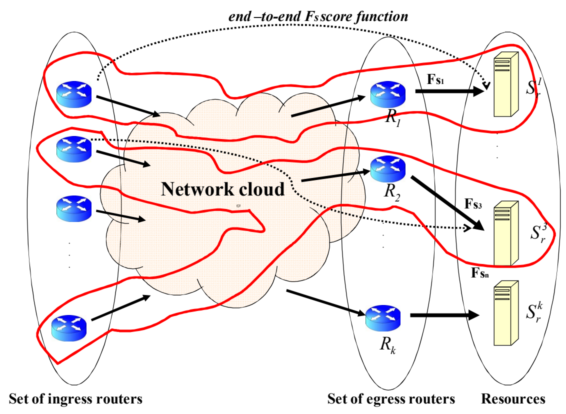


Figure 4: End-to-end shortest path formed between the set of ingress nodes to the available set of resources.

$maxF_s\{\overline{D}(P(R_i,R_i'))^\circ\overline{D}(P(R_i',S_r^k))\} = maxF_s\{\overline{D}(P(R_i,R_i'))^\circ\overline{D}(R_i')\}, \quad \forall\, R_i' \in N_i$. In the above equation, R_i' is any of the N_i set of neighboring nodes of R_i. Therefore, node R_i selects that adjacent node that maximizes its score value, thus pointing to a specific resource. For calculating the distance vector cost of the next to R_k node, we used the operator ° for handling multi dimensional vectors. That operator defines the logical functions (*MIN, MAX, ADD,* etc) between the vector entries.

IV. Score Functions and path (island) Selection

There have been many research efforts on how to combine metrics of different nature for path computation purposes. Generally, the problem of finding a path, subject to constraints on two or more additive and multiplicative metrics in any possible combination is NP-complete [8]. Clearly, there has not been yet proposed a method either to convert the multi-cost to single cost routing problem as in physical layer impairment aware routing algorithms using analytically calculated Q-factors or truly handle the problem as a multi-cost routing problem. We argue in this paper that constraints must be handled based on joint network and non-network resource optimization e.g., minimum number of hops, minimum usage of electronic regeneration, load balancing, etc. For example, it is quite unreasonable a service request with certain SLAs to have a hop count constraint as well. Usually, end-user requirements are bandwidth, delay, server capacity (in terms of CPU cycles, number of subscribed users, etc). On the other hand, network operators would be more interested to balance network load or decrease control overhead. Therefore, combining constraints of different nature may result in misleading results. To this end, we propose in this paper the use of classical optimization algorithms to solve the problem. Among them, meta-heuristics (evolutionary algorithms, ant colony, etc) are the only one that can truly handle a multi-objective optimization problem but however are too complex for path selection especially for a small set of candidate paths. This is because, each core node usually has a small number of neighboring nodes. On the other hand, traditional methods of converting the multi-objective to single-objective optimization path selection are more suitable. Such methods include the *weighted sum* and the ε- *Constraint* approaches. In the first case, a single-objective model is created by weighting *n* objective functions. For example, a candidate score function from node R_i to S_r^k resource could be:$F_{S_r^k} = a_1 B_{R_i\to S_r^k} + a_2 h_{R_i\to S_r^k}^{-1} + a_3 D_{R_i\to S_r^k}^{-1}, \dots, a_k CPU_{S_r^k}$ subject to $\sum_{i=1}^{k} a_i = 1$ and $a_i \in [0,1], i = \{1 \dots k\}$
Optimization function F_s can be modified to represent a usable absolute number by using for example the % deviation over the minimum-maximum range value as follows:

$$F_{S_r^k} = a_1 \frac{B_{R_i} - \mathrm{B}_{\min}}{\mathrm{B}_{\max} - \mathrm{B}_{\min}} + a_2 \left(\frac{h_{R_i\to S_r^k} - h_{\min}}{h_{\max} - h_{\min}}\right)^{-1} + a_3 \left(\frac{D_{R_i\to S_r^k} - D_{\min}}{D_{\max} - D_{\min}}\right)^{-1}, \dots, + a_k \frac{CPU_{S_r^k} - \mathrm{CPU}_{\min}}{CPU_{\max} - CPU_{\min}}$$

where the min, max refer to the corresponding min, max values among the set of all candidate islands in the vicinity of R_i. The other traditional method, termed as ε-*Constraint* optimizes a single cost only, while the remaining ones are not optimized but inserted as constraints in the model. For example, the score function at node R_i when selecting a path toward the different S_r^k service islands could be: $\max\left\{CPU_{S_r^1}, CPU_{S_r^2}, \dots, CPU_{S_r^k},\right\}_r^k$, with the bandwidth, number of hops and delay as constraints: $B_{R_i\to S_r^1}, B_{R_i\to S_r^2}, \dots, B_{R_i\to S_r^k} \geq B, h_{R_i\to S_r^1}, h_{R_i\to S_r^2}, \dots, h_{R_i\to S_r^k} \leq H, D_{R_i\to S_r^1}, D_{R_i\to S_r^2}, \dots, D_{R_i\to S_r^k} \leq D$. In the case of a tie, node R_i selects randomly which island to join. Both schemes are simple but possess some advantages. First, their computational complexity is small and thus can be easily adopted/implemented in a distributed fashion as SO-SOON proposes. Secondly, it is the network provider together with service provider that will decide which constraints are important to them and thus define coefficients in the weighted sum method or which cost to optimize in the ε-*Constraint* method. Figure 5 and Figure 6 show an example of a grid network and how network nodes are organized after routing table establishment. For the self-organization process a combination of the above approaches was used as follows:

$F_s = 50\% \frac{B_{R_i} - \mathrm{B}_{\min}}{\mathrm{B}_{\max} - \mathrm{B}_{\min}} + 50\% \frac{CPU_i - \mathrm{CPU}_{\min}}{CPU_{\max} - CPU_{\min}}$, subject to $D \leq 80 \cdot h, h \leq 3, ST \geq 40\ (a.u)$

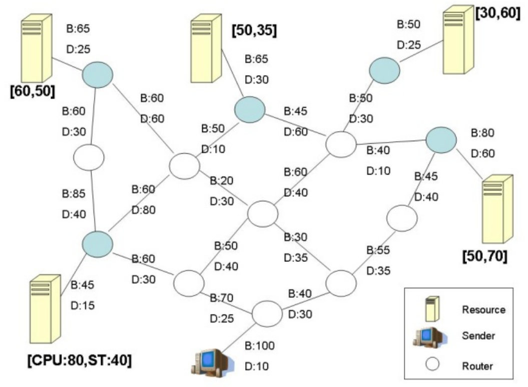


Figure 5: Grid network example consisting of five (non-network) resources and thirteen networking nodes. Numbers on links and resources denote the available bandwidth, delay, CPU and storage respectively.

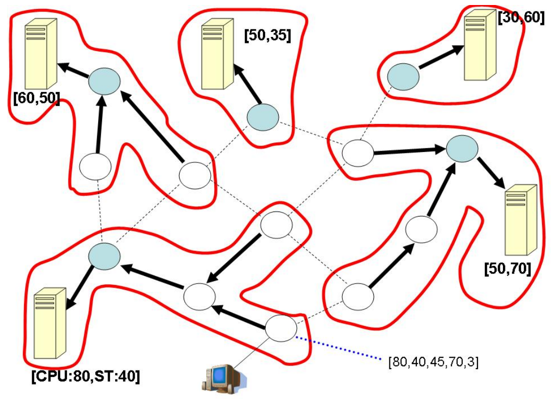
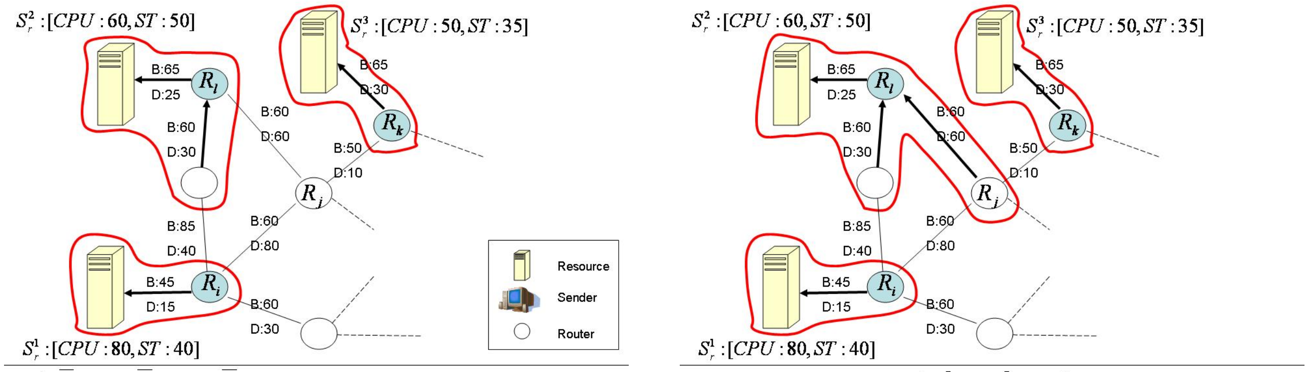


Figure 6: Network example after routing table establishment.



$$S_r^1 : \bar{D}_1(R_j) = \bar{D}(R_{j\to i}) \circ \bar{D}(R_i) = [80,40,45,95,2]$$
$$S_r^2 : \bar{D}_2(R_j) = \bar{D}(R_{j\to l}) \circ \bar{D}(R_l) = [60,50,60,85,2]$$
$$S_r^3 : \bar{D}_3(R_j) = \bar{D}(R_{j\to k}) \circ \bar{D}(R_k) = [50,35,50,40,2]$$

$$\left\{\begin{array}{l} F_s\left[\bar{D}_1(R_j)\right] = 0.5 \\ F_s\left[\bar{D}_2(R_j)\right] \approx 0.67 \\ F_s\left[\bar{D}_3(R_j)\right] = 0 \end{array}\right\}$$

Figure 7: Graphic illustration of an extension of a service island with the addition of node R_j (left figure). Node R_j receives three state packets from R_j,R_l and R_k nodes and calculates the relevant distance vector costs (left figure). R_j decides to point to S_r^2 (right figure) and enters R_l as the next hop on its shortest path route.

Such a score function balances network and server capacity. Figure 7 graphically shows how a service island is extended with the addition of a node. In Figure 7, node R_j receives three state packet from R_i, R_l and R_k nodes, calculates the distance vector costs and adds R_l as the next hop on its shortest path route towards S_r^2 resource.

V. GMPLS Extensions for SO-SOON Implementation

SO-SOON concept can be implemented with currently proposed GMPLS extensions. We will make use of the concept of *forwarding adjacency* proposed by K. Kompella et al. [9]. The concept of forwarding adjacency was proposed to aggregate label switched paths (LSPs) by creating a hierarchy of such LSPs. Routing protocols like open-shortest-path-first (OSPF) floods information for FA links as for normal TE links and thus each Label Switching Router (LSR) may keep FA LSPs in its link state database along with its TE information. A *forward adjacent* LSP is treated as normal TE links by LSRs. To make use of FA notion in SO-SOON scheme, FA-LSPs must be setup dynamically between more than two GMPLS nodes. In that case, each FA-LSP is initiated from the first node (egress router), adjacent to a resource. The newborn FA-LSPs (single link), can be created statically whereas we may assume that there exists an external mechanism (software) to pass performance metrics (i.e CPU availability etc) of that resource. These non-network metrics, together with networking ones must be communicated upstream to extend the service specific FA-LSPs. This is done by network layer protocols like OSPF-TE that keep and flood link state information. To this end and according to [9], the attributes of a FA-LSP are inherited from the LSP that induced its creation. Thus, TE information may propagate with the extension of the FA-LSP. During that process, LSR treats FA-LSPs similar to other TE links and compute which forwarding adjacency to join in a similar way with TE links for path computation. To this end, each LSR node may autonomously decide to join (or not) a FA-LSPs, thus extending a "*service island*".

It must be noted here that in the case of SO-SOON approach, an LSR has only FA-LSPs in his database on a per service basis and this set of FA-LSPs correspond to the number of resources of that specific service in the network.

Each upstream LSR can then perform path computation and decides which FA-LSP to join and thus associate the destination IP address (non-network resource or egress router) of the path message to the computed next hop for that path message. When the service island splits towards different ingress nodes, then the FA-LSP also split to sub-FA-LSPs. In other words, if there exists more than one FA-LSP that originate from different LSR and end to the same LSR, then that common LSP multiplex the two FA-LSPs into a single FA using the concept of Link Bundling. Upon island re-organization and path re-computation, the node may join other FA-LSP, thus directing new service requests to other resources. In that case, ongoing flows continue to be forwarded to the old destination via normal OSPF routing. The old FA-LSPs are not torn down but may start decreasing in size (detaching links), thus ending with a single link between the resource and egress router. Information may continue to be upstream advertised with no upstream node joining it.

VI. Conclusions

In this paper, we have presented a self-organizing service oriented optical network (SO-SOON) architecture, employing self-managed islands of service transparency. We have shown, how networking peers (nodes) are organized or re-organized to form autonomous service entities. The proposed scheme relies on multi-dimensional path selection algorithms and use service proxy to resolve service addressing and QoS issues.

VII. References

[1] C. Prehofer and C. Bettstetter, “Self-Organization in Communication Networks: Principles and Design Paradigms”, IEEE Comm. Magazine, Volume: 43, Issue: 7, page(s): 78- 85, July 2005.

[2] Li, C.-S., Ramaswami, R.: Automatic Fault detection, isolation, and Recovery in Transparent All-Optical Networks. Journal of Lightwave Technology 15(10), 1784–1793 (1997)

[3] Y. Jae-Seung, O. Tonguz and G. Castanon, “Security in All-Optical Networks: Self-Organization and Attack Avoidance”, in proc of ICC 2007, pp. 1329-1335, June 2007.

[4] S. Dixit, E. Yanmaz and O.K. Tonguz, “On the Design of Self-Organized Cellular Wireless Networks”, IEEE Comm. Mag. vol. 43, issue 7, pp: 86–93, July 2005.

[5] R. Dutta, et al. “The SILO Architecture for Services Integration control and Optimization for the Future Internet”, in proc. of IEEE International Conference on Communications, pp. 1899 – 1904, June 2007.

[6] G. Zervas, et al. “Multi-domain optical networks: issues and challenges - Phosphorus grid-enabled GMPLS control plane (GMPLS): architectures, services, and interfaces”, IEEE Comm. Mag., Volume 46, Issue 6, Page(s):128 – 137, June 2008.

[7] Ioannis Tomkos, “Dynamically reconfigurable transparent optical networking based on cross-layer optimization”, 9th Int. Conf. on Transp. Opt. Net., vol. 1, pp. 327 – 327, July 2007.

[8] A. Jukan and G. Franzl, “Path Selection Methods with Multiple Constraints in Service-Guaranteed WDM Networks”, IEEE/ACM Transactions on Networking, 12(1):59–72, Feb. 2004.

[9] K. Kompella, Y. Rekhter, “Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)” RFC 4206, Oct. 2005.