An agent-based distributed protocol for resource discovery and allocation of virtual networks over elastic optical networks

DANILO BÓRQUEZ-PAREDES[1,*], ALEJANDRA BEGHELLI[2], ARIEL LEIVA[3], NICOLÁS JARA[4], ASTRID LOZADA[4], PATRICIA MORALES[4], GABRIEL SAAVEDRA[5], AND RICARDO OLIVARES[4]

[1] *Faculty of Engineering and Sciences, Universidad Adolfo Ibañez, Viña del Mar, Chile*
[2] *Optical Networks Group, Department of Electronic and Electrical Engineering, University College London, UK*
[3] *School of Electrical Engineering, Pontificia Universidad Católica de Valparaíso, Chile*
[4] *Department of Electronic Engineering, Universidad Técnica Federico Santa Maria, Valparaiso, Chile*
[5] *Department of Electronic Engineering,Universidad de Concepcion, Chile*
[*] *danilo.borquez.p@uai.cl*

Compiled November 30, 2022

Network virtualisation is a key enabling technology for "Infrastructure as a Service" provisioning, increasing the flexibility and cost savings offered to customers. By extending the concept of server virtualisation to the network infrastructure, the allocation of different, independent virtual networks over a single physical network is carried out on demand. A fundamental challenge in network virtualisation systems is to choose which physical nodes and links to use for hosting virtual networks into the physical infrastructure, known as the "virtual network allocation" problem.

All virtual network allocation proposals on elastic optical networks assume a centralised operation, deploying a single node with access to the network state global information and assigning resources accordingly. However, such configuration might exhibit the inherent problems of centralised systems: survivability and scalability.

In this paper we present a distributed protocol for resource discovery, mapping and allocation of network virtualisation systems. The distributed protocol is generic enough as to be used with different substrate networks. However, in this article it has been adapted to work over an elastic optical network infrastructure, where further considerations regarding the spectrum continuity and contiguity constraints must also be taken into account.

The distributed protocol is based on the concept of alliances: upon the arrival of a virtual network request, agents located in the physical network nodes compete to form the first alliance able to host the virtual network. Because the first alliances to be formed are also the ones composed by nearby nodes, a good network resource usage is achieved. The feasibility of the distributed protocol was studied by evaluating its ability to successfully establish virtual networks within acceptable time and with low bandwidth consumption from the coordination messages.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

Network virtualisation consists of several virtual networks coexisting over a common physical substrate. It is a crucial technology to implement the idea of "Infrastructure as a Service" (IaaS) [1]. In IaaS systems, customers (also known as tenants) pay for computing and network infrastructure as they need them and the infrastructure provider is in charge of maintaining such

physical resources. Tenants can pay for simple services as a single virtual machine with specific computing capacity or more complex ones as an entire virtual network, where an arbitrary topology must interconnect a set of nodes. Tenants are unaware of the location and the specific hardware/software characteristics of the resources being used [2] and have the flexibility of requesting an increase or decrease of resources according to the dynamics of their needs [3].

Initially, the idea of network virtualisation was proposed as a way to face Internet ossification [4, 5], testing new network protocols in real production settings rather than small testbeds. However, the application of virtual networks has now expanded to new directions, such as wireless networks (including modern 5G networks), fiber-wireless (FiWi) access networks, and optical data center networks (ODCN) [6, 7]. A commercial network virtualisation platform is presented in [8], with a set of implementation strategies and features.

In a nutshell, network virtualisation systems are composed of two main parts: a physical substrate and a set of independent virtual networks established and released on demand over the physical network. The physical substrate, in charge of the actual data processing and transmission, is made of nodes interconnected by bidirectional links. Virtual networks are also made of nodes and links embedded in the physical nodes and links. Each virtual node must be allocated to a physical node with enough capacity to host it. Each every virtual link must be mapped into a physical path.

In virtual networks, the specific meaning of node and link capacities depends on the characteristics of the substrate network considered. In the context of optical networks, some previous works assume physical nodes are optical switches in charge of directing optical signals from one input port to a given output port. In this case, the capacity of nodes can be measured in terms of the number of switching matrices or ports/transceivers, as done in [9–11]. In other cases, where a pair optical switch-electronic node is considered (for example, cloud sites connected by the optical transport network), the node capacity usually refers to the processing capacity or memory storage of the electronic node, as in [12, 13]. Analogously, depending on whether the substrate network is a fixed-grid or a elastic optical network, link capacity might refer to the number of wavelengths [9] or number of spectrum slots [10, 12, 13]. In this paper, we will refer to the node capacity as cloud capacity. That is, processing capacity or memory storage. Please, notice that the concept could also be used in generic terms, since the structure of the virtual network allocation algorithm proposed here does not change with the type of node considered. In terms of link capacity, constraints associated to the capacity of links vary depending on the substrate network. For example, fixed-grid optical networks need to meet the continuity constraint and elastic optical networks need to meet the continuity and contiguity constraints. In any case, the resource allocation must check the suitability of a candidate path, considering the relevant constraints.

Figure 1 shows a schematic of a network virtualisation system over an elastic optical network. Elastic optical networks divide the spectrum of optical communications in frequency slots of 12.5 GHz or 6.25 GHz [14](represented by the squares next to the physical substrate links in the figure) and efficiently allocate spectrum according to the bitrate of connections. By doing so, they can significantly improve the bandwidth utilisation of the optical substrate compared to fixed-grid optical networks [14–16]. Given the ever-increasing data traffic exhibited by current networks, elastic optical networks are thus considered one of the

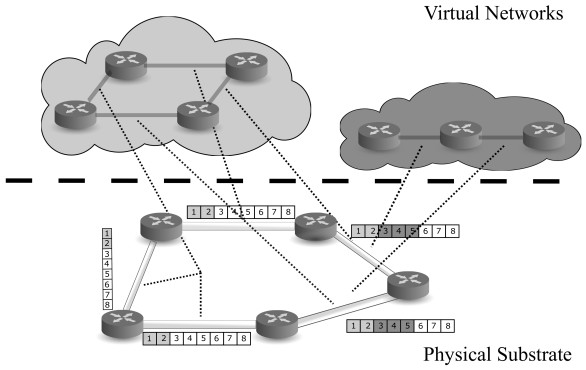


Fig. 1. Virtual network allocation

best alternatives for future optical networks. Thus, in this work, we assumed an elastic optical substrate network considering the continuity and contiguity constraints into the resource allocation process.

Figure 1 illustrates the mapping of two virtual networks over an elastic optical physical substrate made of five nodes and five bidirectional links. Each physical link has a capacity equal to 8 frequency slot units (FSUs). The upper section shows two virtual networks established in the physical substrate: a 4-node ring and a 3-node bus. It is assumed that the amount of FSUs required by each virtual link of the ring and the bus topologies is equal to 2 and 3 FSUs, respectively. The dotted lines represent the allocation of virtual links to physical links, considering the continuity and contiguity constraints of elastic optical networks. Each virtual link has been mapped to a physical link, except for one virtual link in the 4-node ring that has been mapped to a physical path (a sequence of physical links), because the corresponding virtual nodes have been mapped into non-adjacent nodes in the physical substrate. Both virtual networks can have virtual links established in the same physical link (as long as there is enough capacity in the physical link to serve both virtual links). For the sake of clarity, the mapping of virtual nodes is not made explicit in the figure, but it can be deduced by identifying the physical nodes at the extremes of the physical links hosting the corresponding virtual links.

A critical challenge in implementing network virtualisation systems is that of resource discovery, mapping and allocation of resources to the different virtual networks. That is, deciding which physical node/link will host a specific virtual node/link, considering the network state and the constraints imposed by the physical network. To date, most proposals considering an optical substrate assume global network knowledge through a centralised implementation [9, 10, 12, 17–20] which is preferred in small networks. For instance, considering a Software-Defined Network (SDN) architecture or hybrid models relying on GMPLS RSVP-TE and OSPF-TE protocols. [10, 20]. However, centralised systems do not scale well and exhibit survivability problems at the control plane level [7, 21].

Regarding scalability, the request processing capacity of a centralised system is limited by the computer architecture of the central node and the computational complexity of the centralised resource allocation algorithm. The simplest centralised algorithms [17] grow linearly with the number of physical and

virtual nodes, the number of virtual links and the physical link capacity (number of slots). Hence, big size substrate networks with increased spectrum capacity (something very probable with the emergence of multiband elastic optical networks [22]) dealing with big virtual network topologies will naturally require longer running times. This scenario combined with a highly dynamic environment, where virtual network requests arrive faster than the central node request processing speed, will lead to requests being blocked due to central node overload rather than due to lack of network capacity. In terms of survivability, a failure in the central node in charge of the resource allocation task renders the entire network inoperative. Usually, the solution to this failure situation consists on adding a backup central node. Under normal conditions, such backup node might also be used as a second processing unit, to also improve scalability. However, this adds to the cost of the system. Additionally, centralised systems may not react fast enough to recover from network components failures.

Distributed approaches overcome the scalability and control plane survivability issues of centralised systems. In terms of scalability, by distributing the request processing among the network nodes, a faster overall request processing can be achieved. On one side, nodes execute computationally simpler tasks than those executed by a central node. On the other, different requests can be processed on different parts of the network. Regarding survivability, a failure in any of the network nodes in charge of the resource allocation task only affects the participation of that node and its adjacent links in the allocation task, without stopping the whole allocation process from progressing in other parts of the network. Additionally, failures are quickly detected (and eventually recovered) by local nodes. However, a distributed approach might become unfeasible because of the time required to coordinate the discovery and allocation processes, the obsolescence of information, and the amount of bandwidth used to exchange coordination messages.

This paper presents a distributed protocol for resource discovery and allocation in a network virtualisation system over elastic optical networks. To the best of the authors' knowledge, this is the first time that a distributed protocol is proposed for this type of substrate. Although there are a few previous works presenting distributed approaches in generic substrates, for example [23–27], they are either not fully distributed (relying on delegation nodes to perform the virtual network allocation task [24, 27]), concentrate on just one aspect of the allocation process (as [26], that focuses only in the virtual link mapping stage), assume unlimited capacity in the substrate network [23] or use strategies that result in virtual nodes being mapped in far away physical nodes [24, 25, 27], with the consequent inefficient use of link capacity. In contrast, our proposal is fully distributed, covers the 3 stages of discovery, mapping and allocation of resources, considers the limited capacity of physical resources and use a cooperative strategy that leads to virtual nodes being mapped in neighbour nodes.

We study the feasibility of the proposed protocol in terms of its ability to establish virtual networks, the time it takes to do so and the amount of messages exchanged to perform the discovery, mapping and allocation tasks. Its performance is also compared to that of a centralised approach.

The rest of this paper is structured as follows: Section 2 presents the network models and the problem statement. Section 3 presents an overview of the distributed protocol, whilst Section 4 describes the behaviour of the agents. Section 5 presents the performance evaluation of the protocol in terms of its ability to establish virtual networks and the time and number of messages required to discover and allocate resources. Section 6 concludes the paper.

2. NETWORK MODELS AND PROBLEM STATEMENT

A. Physical network model

A physical network is modelled as a graph $P = (NP, LP, CP_n, CP_l)$, where NP and LP represent the set of physical nodes and links, respectively; CP_n and CP_l represent the capacity of each physical node and link, respectively. The capacity of a physical node can correspond to different hardware components, depending on what type of node is considered. Given that the algorithm proposed here is easily adaptable to any of those situations, we will use the term "node capacity" in a generic way from now on. Link capacity is expressed in terms of frequency slot units.

B. Virtual network model

A virtual network is modelled as a graph $V = (NV, LV, CV_n, CV_l)$, where NV and LV represent the set of virtual nodes and links, respectively. CV_n and CV_l represent the capacity required by each virtual node and link, respectively.

C. The virtual network allocation problem

The virtual allocation task consists on finding a set of physical nodes and links where the current virtual network request can be served. Each virtual node must be hosted by a different physical node. Each virtual link must be transformed to an optical path connecting the physical nodes where the virtual nodes at the extreme of the virtual link have been hosted.

A virtual network establishment request is defined by the tuple (VN_{ID}, CV_l, A, CV_n), where VN_{ID} is the virtual network ID, CV_l is the number of slots to be allocated to each virtual link, A is the adjacency matrix describing the topology of the virtual network, and CV_n is a number representing the capacity required by each virtual node, such as the processing capacity or memory required.

The objective of the virtual network task is maximising the number of virtual networks established, subject to the node and link capacity constraints described by Equations 1 and 2 below.

$$CP_n \geq \sum_{\forall j | VN_j \in B_n} CV_n^j \qquad ,\forall n \in NP \qquad \textbf{(1)}$$

$$CP_l \geq \sum_{\forall j | VN_j \in D_l} CV_l^j \qquad ,\forall l \in LP \qquad \textbf{(2)}$$

where CP_n and CP_l represent the capacity of a physical node and a physical link, respectively, CV_n^j and CV_l^j are the node capacity and the link capacity of the j-th active virtual network, and B_n and D_l are the set of active virtual networks using capacity of physical node n and physical link l.

In the case of an elastic optical network, the contiguity and continuity constraints must also be met. That is, the frequency slots must be contiguous and occupy the same spectral position in each link of the path.

D. Virtual network release

In fully dynamic environments, virtual networks have a limited holding time. In that case, a virtual network release request is defined only by the VN_{ID}. Upon receiving a release request, allocated resources must be freed.

In the following section, a distributed protocol that solves the problem stated above is presented.

3. THE DISTRIBUTED PROTOCOL: AN OVERVIEW

The protocol proposed here is based on an Agent-Based Model (ABM) [28], composed of autonomous individuals (agents) interacting with each other and following local rules to achieve a common goal. Such a common goal is to successfully establish virtual networks, satisfying the capacity and continuity and contiguity constraints of the elastic optical substrate while efficiently using physical resources. The interaction between agents is carried out by exchanging control plane messages, similar to what RSVP-TE or any other signalisation protocol envisaged for elastic optical networks does [18].

In this work, each agent resides in a physical node and has unique ID (a number between 1 and N, the number of physical network nodes) associated with it. The agent stores the capacity of its physical node, the spectrum availability of the links directly connected to its corresponding physical node, and several state variables needed for the resource discovery, allocation, and release processes.

In one of the physical nodes there is a majordomo-like process in charge of receiving virtual network requests, adding them a unique ID and distributing them to all the network nodes. In a SDN-based infrastructure, such majordomo process would be analogous to a SDN Controller, although with diminished capabilities since its task is reduced to send requests to nodes. To do so, it could use, for example, the OpenFlow protocol.

Figure 2 shows a high-level representation of the main actions taken by the distributed protocol when a virtual network establishment request is sent by the majordomo-process to all agents. Agents are in idle state, unless they are activated by either receiving a virtual network establishment request or a virtual network release request from the majordomo. Figure 2a shows a virtual network request arriving to the physical network. The thick arrow represents the request being distributed to all agents. Immediately after receiving a establishment request, agents transit from an idle state to an active state and check their node capacity. Agents residing in a node without enough capacity, disqualify themselves from the allocation process (Figure 2b). Next, each agent with enough capacity aims to form alliances with neighbouring agents to host the virtual network. This is done by exchanging messages between neighbouring agents, as shown in Figure 2c. By coordinating the building of an alliance with neighbouring agents (instead of agents located in distant nodes), the chances of using the minimum possible amount of resources are increased. As agents agree on being allies, alliances are enlarged, as shown in Figure 2d. When an alliance has a high enough number of physical nodes, the allocation process is performed. If successful, the virtual network is allocated (Figure 2e) and agents transit to the idle state again (Figure 2f).

When receiving a virtual network release request (not shown in Figure 2), each agent hosting a node of such virtual network updates the state of its allocated resources to signal that the released resources are now available for future establishment requests.

In the following section, a detailed description of the steps taken by each agent when receiving a request is given.

4. THE DISTRIBUTED PROTOCOL: AGENTS BEHAVIOUR

The distributed protocol works by the coordinated action of each agent in the network. Figure 3 describes the local computation performed by each agent after receiving a virtual network establishment request. It can be seen that each agent first goes through a discovery stage (where neighbours are detected and eventually included in the alliance) followed by an allocation stage (where the node mapping and slot allocation is performed). Each agent has access to their own set of state variables, described in Table 1. The meaning of the state variables will be explained in the following paragraphs, where each agent state evolution during the discovery, allocation, and release processes is described. Thus, the order used to list the state variables in Table 1 is the order they appear in the following paragraphs of this section.

A. Resource discovery stage

During the stage, the agent transits through the states Begin, Vote, Proposing alliance and Synchronisation, described below.

Begin: This is the starting state, where all agents are idle. This is the state they are found when they are not discovering, allocating or releasing resources.

In this state, every agent is the only member of its own alliance, its state variable *Confirmed_allies* only contains the ID of the agent itself and, all other state variables are either empty or equal to zero.

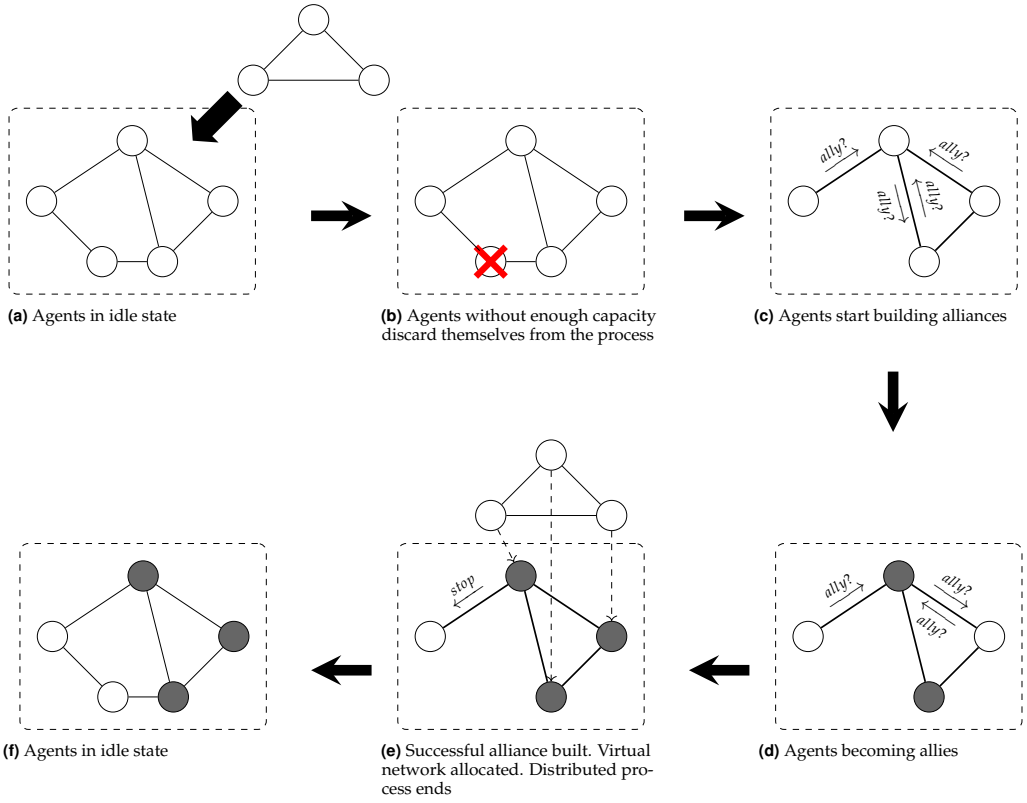
When a new virtual network request is received (Figure 2a), each agent compares the available capacity of its physical node with the capacity required by the nodes of the virtual network (indicated by the parameter CV_n in the request). The available capacity is obtained by subtracting the amount of capacity already used (stored in the state variable *Allocated_nodes*) from the physical node capacity. If the physical node has enough capacity to host a virtual node, then the agent changes its status to "Vote". Otherwise, it remains in the "Begin" state (Figure 2b).

Vote: Immediately after transiting to the "Vote" state, the agents select a leader for their alliance. The leader is simply the node with the lowest ID and is in charge of coordinating the process of alliance building. Since all agents have a list of their allies in the state variable *Confirmed_allies*, the selection of the alliance leader does not need any message exchange.

Once the leader is selected, each ally (an agent that is an alliance member) starts a resource discovery process focused on finding new nodes to join the alliance. That is, the ally selects an adjacent node as a candidate to join its alliance. Adjacent nodes whose IDs are included in the state variable *Forbidden_allies* are not considered in this search. The one connected to the link with the best fitness function value is selected among all allowed adjacent nodes. Such function can be any function that takes into account aspects like the link spectrum utilisation and length. The agent then stores the ID of its candidate node in its state variable *Best_candidate* and sends a [vote] message to the alliance leader.

Figure 4a shows the format of a [vote] message, that is a 9-byte message made of the following fields: the first byte is an 8-bit value used to identify the type of message (00000000, for [vote] messages), the second 4 bytes are used to store an integer number corresponding to the ID of the node connected to the best candidate node and the next 4 bytes are used to store the integer value of the fitness function associated to the candidate node.

If the agent does not find a suitable adjacent node to join



(a) Agents in idle state
(b) Agents without enough capacity discard themselves from the process
(c) Agents start building alliances
(d) Agents becoming allies
(e) Successful alliance built. Virtual network allocated. Distributed process ends
(f) Agents in idle state

Fig. 2. Overview of the processing of a virtual network establishment request by the distributed protocol

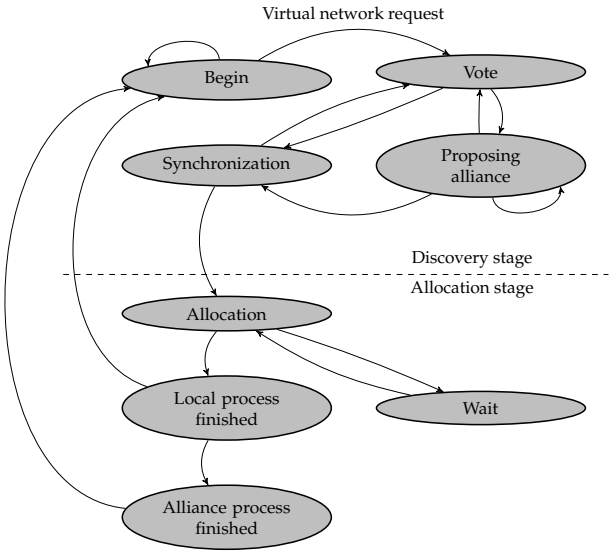
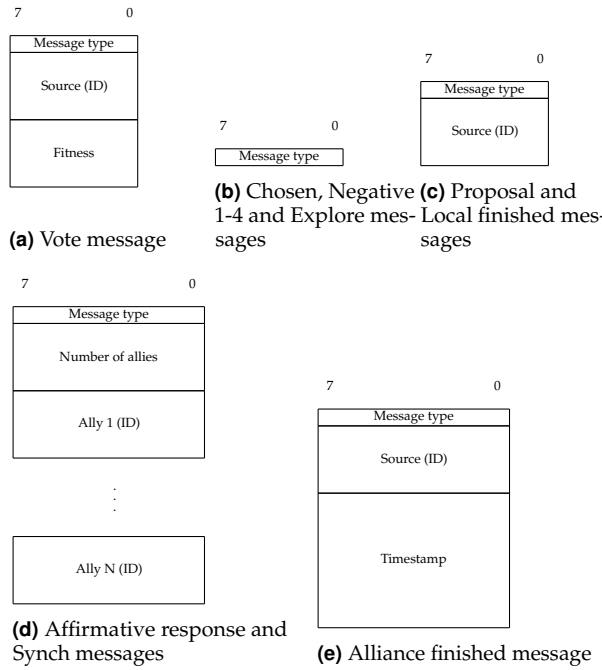


Fig. 3. Agent state diagram



(a) Vote message
(b) Chosen, Negative 1-4 and Explore messages
(c) Proposal and Local finished messages
(d) Affirmative response and Synch messages
(e) Alliance finished message

Fig. 4. Protocol messages format

State variable name	Description
Confirmed_allies	List of physical nodes IDs that are already part of the alliance.
Allocated_nodes	List of 2-element vectors. Every duple contains the ID of a virtual network that has successfully established a virtual node in the node the agent resides and the node capacity allocated to that virtual node.
Forbidden_allies	List of IDs of physical nodes that cannot be selected to be part of the alliance.
Best_candidate	Integer variable that stores the identification of the physical node selected to be part of the alliance.
Votes_received	Integer variable that stores the number of votes received.
Chosen	Boolean variable that indicates if the ally has been chosen to make the connection with the new ally.
Node_finished	Boolean variable that indicates if the node has finished its process.
Allocating_link	Boolean variable. Equal to True if the node is in the process of allocating link resources.
Wait_time	Integer variable that stores a random value corresponding to the time that the node must wait in case of deadlock.
Alliance_finished	Boolean variable that indicates if the alliance has finished its allocation process.
Allocated_links	List of 4-element vectors. Every 4-tuple contains the ID of a virtual network, the ID of a link, the first slot of the link allocated to the virtual network, and the number of slots allocated.

Table 1. State variables of each router

the alliance (this happens when all adjacent nodes are already members of this alliance or in the states "Local process finished", "Allocation", "Wait" or "Alliance process finished"), it sends a null [vote] message (all fields equal to zero) to the leader.

As the leader receives the [vote] messages, its state variable *Votes_received* increases accordingly. Once all [vote] messages are received, the leader selects the candidate node with the best fitness value and sends a [chosen] message to the ally that proposed that candidate node. Then, the leader resets its state variable *Votes_received* to 0. Figure 4b shows the format of a [chosen] message: just a 1-byte field to store the message ID (equal to 00000001 for a [chosen] message). Upon receiving a [chosen] message, the ally changes its state variable *Chosen* to 1 and transits to the "Proposing Alliance" state, where it sends a [proposal] message to its candidate node to propose it to join the alliance (Figure 2c). Figure 4c shows the format of a [proposal] message: a 5-byte message with two fields. A 1-byte field storing the message ID (equal to 00000010 for a [proposal] message) and a 4-byte field to store an integer number corresponding to the ID of the node sending the message. The allies that are not chosen to send the [proposal] message remain in the "Vote" state, waiting for a [synch] message.

Depending on the response message received from its candidate node, the ally can transit to different states:

- The reception of a [negative 1] message results in the ally remaining in the "Proposing Alliance" state, where it keeps on sending a [proposal] message and waits for an eventual acceptance due to eventual changes in the candidate node state. As shown in Fig. 4b, a [negative 1] message is a 1-byte message storing the message ID (equal to 00000011 for a [negative 1] message).

- The reception of a [negative 2] message means the candidate node will not join this alliance at any point of the allocation process (e.g., because of capacity exhaustion). As shown in Fig. 4b, a [negative 2] message is a 1-byte message storing the message ID (equal to 0000000100 for a [negative 2] message). The reception of a [negative 2] message leads the ally to include the ID of the candidate node to the list *Forbidden_allies* and then transit to the "Vote" state, where the whole process of selecting a new candidate node is started again.

- The reception of an [affirmative] message makes the ally and the candidate node change their state to "Synchronisation". An [affirmative] message can only be received from a node that has been, in turn, sending [proposal] messages to the ally. As shown in Fig. 4d, an [affirmative] message is a 4N+1-byte message (N: number of physical nodes) made of the following fields: a 1-byte field used to identify the type of message (equal to 00000101 for an [affirmative] message) and 4N bytes to store the ID of the current members of the alliance.

Figure 5 shows an example of two incomplete alliances executing the resource discovery process. In Alliance 2, agent 1 (Node ID 1 in the figure) is chosen to send a proposal to agent 4 in Alliance 1 to merge both alliances. When agent 4 receives the proposal, it executes its own discovery process to find its best candidate to join Alliance 1. Thus, it replies with a [negative 1] message, and thus, agent 1 keeps sending [proposal] messages. When agent 4 finishes its discovery process, it sends its [vote] message (Node with ID 1) to its alliance leader (agent 3, the node

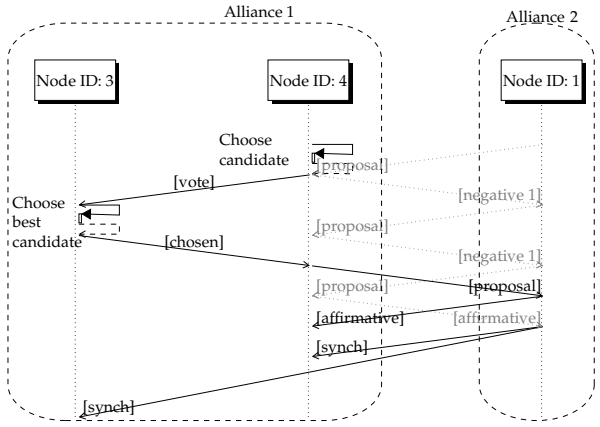


Fig. 5. Discovery sequence diagram

with the lowest ID in Alliance 1). Agent 3 processes the votes and selects the best candidate node to join the alliance. Such node is Node with ID 1. Thus, agent 3 sends a [chosen] message to agent 4. Finally, agent 4 sends a proposal to the best candidate, receiving an affirmative response.

Synchronisation: In the "Synchronisation" state the ally and its candidate node exchange information about the members of their alliances and these alliances are merged (Figure 2d). To do so, the node with the lowest ID between them (agent 1 from Alliance 2 in the example of Figure 5) sends a [synch] message to all members of the new alliance with a list of the new members. As shown in Fig. 4d, a [synch] message is a 4N+1-byte message (N: number of physical nodes) made of the following fields: a 1-byte field used to identify the type of message (equal to 00000110 for a [synch] message) and 4N bytes to store the ID of all the members in the alliance. Upon receiving this message, every ally updates its state variable *Confirmed_allies* to include the new members.

Once an alliance has a number of allies greater than or equal to the number of virtual nodes required by the virtual node request, all allies change their status to "Allocation". Otherwise, all allies transit to "Vote" again.

B. Resource allocation stage

During this stage, the agents of an alliance with enough physical nodes perform the node mapping and spectrum allocation (states Allocation and Wait). On successful achievement of these tasks, they transit to Local process finished and Alliance process finished.

Allocation:In this state, each ally rejects any [proposal] message with a [negative 2] response, as the alliance has now a high enough number of nodes. Additionally, each ally attempts to allocate the resources required by the virtual network (processing units in the physical node and number of contiguous slots in the corresponding physical links).

First, each ally sorts the ID of its alliance members (including itself) from lowest to largest. Next, it identifies its position in the ordered ID list. The physical node in the i-th position will be hosting the i-th virtual node included in the virtual network request. Thus, the state variable *Allocated_nodes* is updated by adding a 2-vector storing the ID of the virtual network and the capacity allocated to it in this node. If the alliance has more nodes than the number required by the virtual network request, the nodes not being assigned to a virtual node change their status to "Local process finished", update their state variable *Node_finished* to True, and send a [local finished] message to its leader. As shown in Fig. 4c, a [local finished] message is a 2-byte message storing the message ID (equal to 00001010 for a [local finished] message) and a 4-byte integer field to store the ID of the node. Once the agent knows what virtual node will be allocated to its physical node, the process of allocating the spectrum resources required by its adjacent virtual links start by changing the state variable *Allocating_link* to 1. The agent obtains the information of what nodes need to be connected from the adjacency matrix A contained in the request.

To avoid replicated allocation of spectrum resources by the virtual nodes at the extremes of the same virtual link, only the node with the lowest ID among them can start the link allocation process. The node with the highest ID sends a [local finished] message to the alliance leader and does not participate in the link allocation process. The lowest ID node sends an [explore] message to the destination node (connected at the other extreme of the virtual link) using a pre-computed path. Upon receiving the [explore] message, each physical node along such pre-computed path changes its state variable *Allocating_link* to True and updates the link slot availability information. If there are not enough slots in one of the links, a [negative 3] message is sent back to the ally, and the state variable *Allocating_link* is changed back to False in every node receiving the [negative 3] message along the backward route. As shown in Fig. 4b, a [negative 3] message is a 1-byte message storing the message ID (equal to 00000111 for a [negative 3] message).

If the [explore] message arrives successfully at the destination node, the agent in this node sends an [affirmative] message back to the agent in the source node. Upon receiving the [affirmative] message, every agent along the route allocates the corresponding spectrum resources and changes its state variable *Allocating_nodes* to False.

When the agent state variable *Allocating_link* is equal to True, the agent rejects any [explore] message asking it to initiate another link allocation process (to establish another virtual link) by sending a [negative 4] message back. As shown in Fig. 4b, a [negative 4] message is a 1-byte message storing the message ID (equal to 00001000 for a [negative 4] message). The physical node receiving the [negative 4] message changes its state to "Wait", where it waits for a random number of clock ticks – stored in the state variable *Wait_time* - before transiting again to the "Allocation" state for a new resource allocation attempt. When a physical node successfully terminates the resource allocation process, it changes its status to "Local process finished" and sends its leader a [local finished] message.

Once the leader receives as many different [local finished] messages as the number of nodes on its alliance, it sends an [alliance finished] message to all the physical network nodes to announce that the alliance has successfully allocated the resources required by the virtual network. As shown in Figure 4e, an [alliance finished] message is made of a 1-byte field used to identify the type of message (equal to 00001100 for an [alliance finished] message), a 4-byte integer to store the ID of the leader sending the message, and an 8-byte integer to store the timestamp representing the moment when the [alliance finished] message was sent. Upon receiving the [alliance finished] message, all the members of the successful alliance change the state variable *Alliance_finished* to 1, and the members of the remaining alliances release eventually allocated resources. After this, all physical network nodes transit to the "Begin" state, signalising they are ready to start a new discovery/allocation process or a release process. For the sake

of clarity, this transition from any state to the "Begin"state is not depicted in Fig. 3.

Wait: To avoid two agents attempting simultaneous allocation of link resources, when an [explore] message arrives at a node that is already in the allocation process (state variable *Allocating_link* equal to True, a [negative 4] message is returned to the sending node. Upon receiving a [negative 4] message, the node changes its status to "Wait". In this state, the sending node waits a random time between 1 and 20 ticks before sending again an [explore] message, thus continuing its assignment. This way of solving the problem is inspired by the IEEE 802.3 standard for Ethernet networks [29].

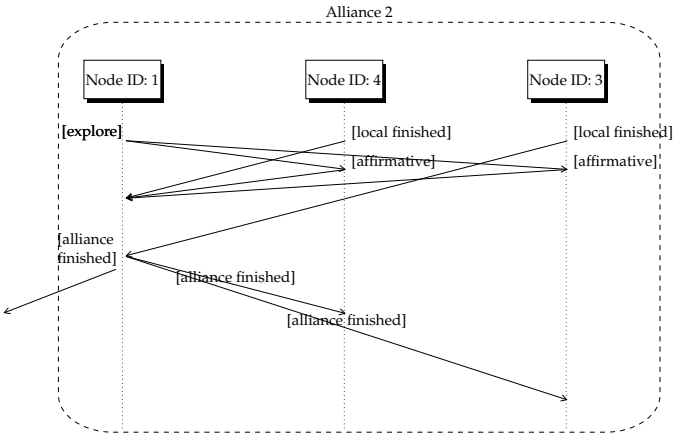


Fig. 6. Allocation sequence diagram

Figure 6 shows an example of the link allocation process of an alliance made of 3 nodes (nodes 1, 3, and 4). Let us assume that a virtual link must be established between the node pairs 1-3 and another between 1-4. Each agent starts the link allocation process: agent 1 (that is also the leader of the alliance) sends an [explore] message to nodes agents 3 and 4, and receives an [affirmative] message from them. After receiving the [affirmative] messages, agent 1 allocates resources on both links connected to agents 3 and 4. Meanwhile, since agents 3 and 4 do not have any virtual link to allocate, they send a [local finished] message to the alliance leader (agent 1), and transit to the "Local process finished state". When the leader receives all its allies' [local finished] messages, it sends an [alliance finished] message to all the network nodes.

Local process finished: An agent transits to this state after sending a [local finished] message to the leader of its alliance. When the leader receives several [local finished] messages equal to the number of allies, it transits to the *Alliance process finished* state.

Alliance process finished: Only alliance leaders transits to this state. At this state, the leaders sends an [alliance finished] message to all the network agents and the majordomo. After sending the [alliance finished], the leader transits to the *Begin* state.

When an agent receives an [alliance finished] message, it will carry out the following steps (depending on whether the message is received from its own leader or not and its state):

- If the [alliance finished] message is received from the leader of the agent's alliance, then the agent transits to the *Begin* state.
- If the [alliance finished] message is received from the leader of another alliance and the agent is the leader with a timestamp greater than the timestamp in the [alliance finished] message, it sends a release request to all its allies (after releasing the resources, these allies transit to the *Begin* state).
- If the [alliance finished] message is received from the leader of another alliance and the agent is not the leader of its alliance, then this agent does nothing.

If, after the arrival of the virtual network request, T_o ticks have been elapsed without a message *Alliance_finished* being sent to the majordomo, the virtual network request is rejected. Such timeout-based control is used (instead of an explicit control mechanism that would need additional messages exchange) to avoid further complexity.

C. Resource release

Figure 7 describes the behaviour of each agent after receiving a virtual network release request from the majordomo process.

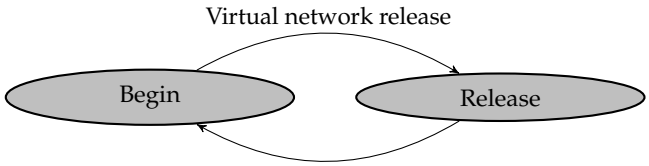


Fig. 7. Agent state release diagram

When an agent receives such a release request, it transits to the Release state. A release request stores the ID of the virtual network to be released. Thus, each agent checks whether such ID is in the list of their state variables *Allocated nodes* and *Allocated links*. If so, the corresponding resources are released by deleting those elements from the lists. After this, the agent transits to the *Begin* state.

Please notice that, although the distributed algorithm has been described in the context of elastic optical networks, it can be easily modified to be applied to different physical substrates. This can be done by changing 3 modules. First, the module in charge of detecting if physical nodes have enough capacity to host a virtual node (action taken in the Begin state) must be adapted to the specific nodes considered. Second, the module responsible for selecting the next ally (using a suitable fitness function in the Vote state). Finally, the module performing the allocation of link resources (Allocation state).

5. CASE OF STUDY

To demonstrate the feasibility of the distributed protocol here proposed, we implemented it using the multi-agent simulation environment Netlogo [30]. In Netlogo, the following configuration was used:

- Physical substrate. The physical substrate is an elastic optical network. The network topology used was NSFNet, composed of 14 nodes and 21 bidirectional links. This topology is the most commonly used in the literature [17]. All physical nodes are equipped with the same capacity (10, 20, 30, 40, or 50 units), and each physical link is equipped with 100 slots. In each physical node, there is an agent governed by the rules described in the previous sections.

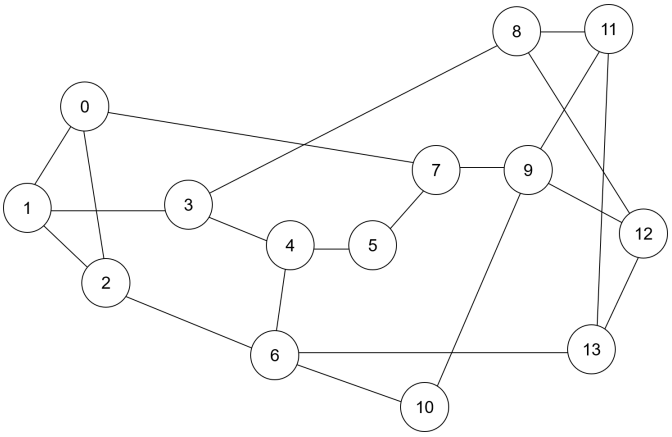


Fig. 8. NSFNet topology

- Virtual networks. Each virtual network is defined by its adjacency matrix A (virtual nodes connected by virtual links) and capacity requirements for virtual nodes and virtual links (number of slots). Analogously to previous work where a limited set of topologies is considered (e.g. [31–33] and [34]), for this case of study, two virtual topologies were considered: buses and rings. Virtual nodes require 1 capacity unit. Virtual links require 2 or 3 spectrum slots that correspond to a bitrate of 40 and 100 Gbps using 8-QAM [35]. Each simulation experiment was executed using the same virtual network topology (e.g., rings of 5 nodes), requesting the same number of slots per virtual link (e.g., 2 slots) and the same capacity in each node (e.g., 1 unit of capacity).

- Traffic generation. Incremental traffic is assumed. That is, once a virtual network is established on the physical substrate, it remains established indefinitely (i.e., virtual networks never request a release of the allocated resources). A new virtual network establishment request is generated by the majordomo only after the previous request has been processed. Requests sent by the majordomo are assumed to be transmitted instantaneously to all the nodes in the network. In this way, all nodes start the execution of the distributed algorithm simultaneously.

- Fitness function. The best candidate to join the alliance is the neighbour node connected to the link with the highest spectrum availability, measured as:

$$A_l = CP_l - \sum_{\forall s \in l} a_s \quad \textbf{(3)}$$

where A_l is the number of slots available in link l, CP_l is the total number of slots of link l (in this case of study, this is equal to 100 for all network links) and a_s is equal to 1 if slot s is used and 0 if not. If two links have the same slot availability, the shortest one is chosen as the best candidate.

- Agent allocation information. The agent has access to a list of pre-computed routes (one for each other node), used together with the first fit algorithm. In addition a single modulation format was available for every connection. All routes are assumed to be under the optical reach determined by the modulation format.

- Simulation time. A simulation experiment ends immediately after the first request reject. That is, when 5000 clock ticks have passed without the system establishing a virtual network. As a reference, the longest time elapsed for a virtual network to be established was approximately equal to 500 ticks in the simulation runs.

- Simulation experiments. For each simulation configuration (defined by the virtual topology, number of capacity units required by virtual nodes and number of slots required by virtual links), 20 different simulation experiments were run.

The performance of the protocol was evaluated in terms of:

- The number of virtual networks successfully allocated as a function of the physical network capacity and virtual network requirements

- The time required to establish a virtual network (resource discovery plus resource allocation time) as a function of the physical network capacity and virtual network requirements

- The number of coordination messages sent by the network nodes as a function of the physical network capacity and virtual network requirements

Algorithm 1. Centralised algorithm

Require: $NP, type, |NV|$.
1: $ranking = list()$
2: $N_c = NP - N_{nc}$
3: **for** $\{n \in N_c\}$ **do**
4: $d = nodal\_degree(n)$
5: $L_n = get\_links(n)$
6: **for** $\{l \in L_n\}$ **do**
7: $used\_slots+ = get\_used\_slots(l)$
8: $length+ = get\_length(l)$
9: $n.utilisation = used\_slots/d$
10: $n.length = length/d$
11: $ranking.add(n)$
12: $sort\_by\_utilisation(ranking)$
13: $chosen\_nodes = top(|NV|)$
14: **for** $\{i \in \mathbb{Z} : i < len(chosen\_nodes) - 1\}$ **do**
15: **if** $slots\_available(path(chosen\_nodes_i, chosen\_nodes_{i-1}))$ **then**
16: $connect(chosen\_nodes_i, chosen\_nodes_{i-1})$
17: **elsereturn** *Non-successful*
18: **if** $type =$ "$ring$" **then**
19: **if** $slots\_available(path(chosen\_nodes_{first}, chosen\_nodes_{last}))$ **then**
20: $connect(chosen\_nodes_{first}, chosen\_nodes_{last})$
21: **elsereturn** *Non-successful*
return *Successful*

By way of comparison, a centralised algorithm, shown in Algorithm 1, was also evaluated. The algorithm receives the set of physical nodes (NP), the type of virtual network (ring or bus) and the number of virtual nodes ($|NV|$) as input arguments and performs 3 steps. First, it ranks the physical nodes (lines 1-12) with enough capacity (the set of nodes without enough capacity, N_{nc} are discarded from further consideration in line 2). For a fair comparison with the distributed protocol, nodes

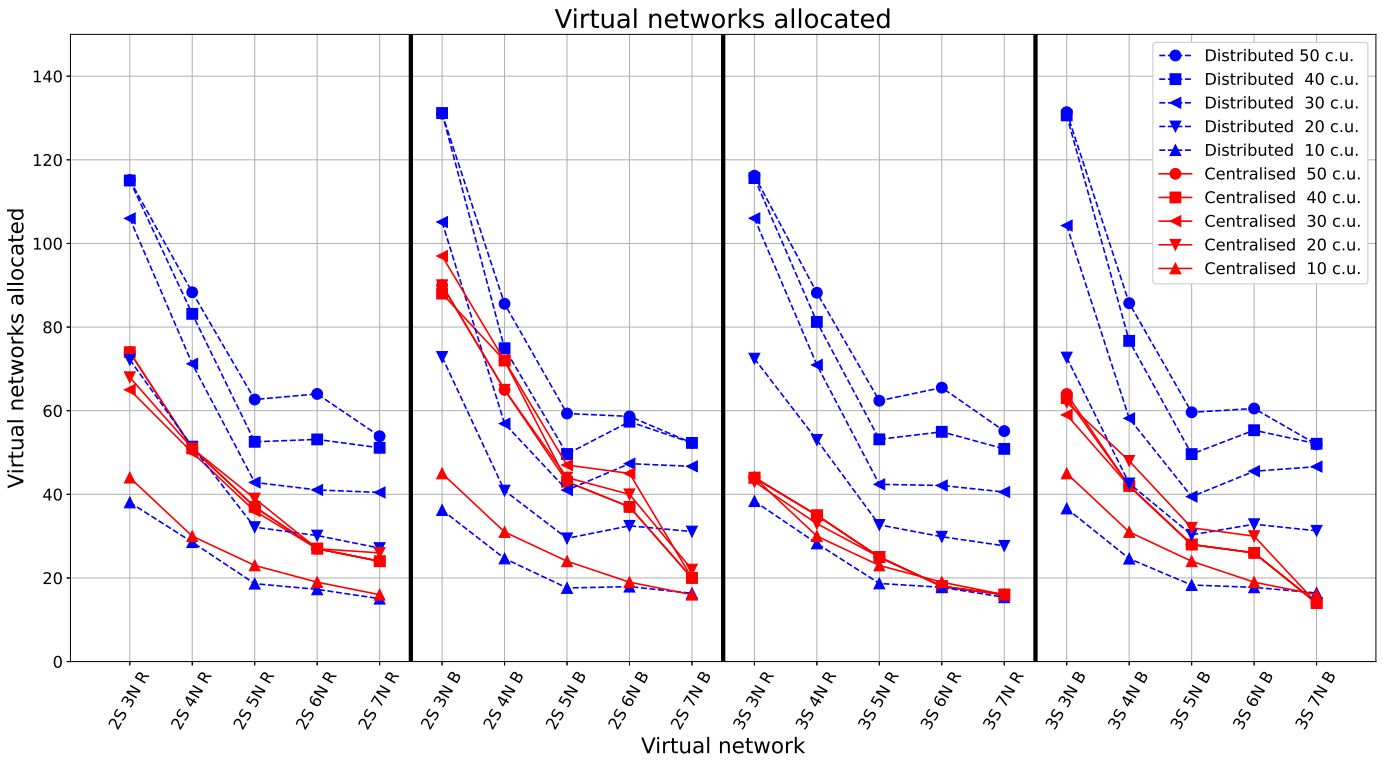


Fig. 9. Virtual networks allocated to physical network

are ranked from highest to lowest slot availability of its adjacent links. Ties are broken considering the length of the links (shorter links preferred). Second, the nodes selected to map the virtual nodes are the first $|NV|$ nodes in the ranking, where $|NV|$ is the number of virtual nodes (line 13). Finally, lines (14-21) are in charge of connecting the physical nodes using the shortest path.

This algorithm has a computational complexity of O($|NP|$*d+$|NP|$*log($|NP|$)+$|NV|$+$|LV|$*S), where $|NP|$, d, $|NV|$, $|LV|$ and S are the number of physical nodes, network nodal degree, number of virtual nodes, number of virtual links and number of slots, respectively. The centralised algorithm was implemented using the C++ library Flex Net Sim [36]. The code for the distributed protocol and the centralised algorithm is available in [37].

Figure 9 shows the number of virtual networks allocated during the execution of the distributed protocol and the centralised algorithm. The number of virtual networks successfully established is one of the critical performance measures of the algorithm, as it impacts the revenue of the network operator and client satisfaction. Each point in the curve corresponds to the average of 20 simulation experiments. Each name in the horizontal axis represents a different virtual network topology. The first two characters correspond to the number of slots used by the virtual network: 2 or 3 slots. The second two characters refer to the number of virtual nodes: $3N$, $4N$, $5N$, $6N$, and $7N$. Finally, the last character indicates the type of virtual topology: "B" corresponds to a bus topology and "R" to a ring topology. Each line represents a different capacity for the physical nodes: 10 , 20 , 30 , 40, and 50 units.

It can be seen that - for the centralised and the distributed approaches - the number of allocated virtual networks increases with the capacity of the physical nodes, as more virtual nodes can be accommodated in the physical nodes. The difference between the distributed curves decreases for 40 and 50 capacity units because the number of virtual networks successfully allocated at these capacities is limited by the physical link utilisation instead of the capacity of the nodes.

The centralised algorithm achieves better performance in low-capacity scenarios, but its performance quickly deteriorates in scenarios with higher node capacities, where it is outperformed by the distributed approach. As more virtual networks are established, the centralised algorithm is becoming affected by its inability to establish the virtual networks using physical nodes in close proximity. As a result, virtual links must be established using longer paths, leading to higher usage of slots per virtual network. This, in turn, leads to a lower number of virtual networks being established.

On the contrary, the distributed protocol works by proximity, as agents attempt to be allies with the near nodes. As a result, nodes form alliances with their neighbours, allocating virtual networks in compact sectors of the physical network. This leads to a lower bandwidth consumption per virtual network and thus, to a higher number of them being established.

A theoretical upper limit for the number of virtual networks allocated avoiding the constraints of finite link capacity is given by the expression $\lfloor \frac{|NV|}{|NV|} \rfloor \cdot \lfloor \frac{CP_n}{CV_n} \rfloor$. In this expression, $|NV|$ and $|NV|$ are the number of physical and virtual nodes, respectively; CP_n is the number of capacity units each physical node is equipped with, and CV_n is the number of capacity units required by each virtual node. For a low node capacity (10 units), where the finite capacity of links does not significantly impact the allocation process, the maximum number of virtual networks given by this expression is equal to 40, 30, 20, 20, and 20 for a number of virtual nodes equal to 3, 4, 5, 6 and 7, respectively. From

Figure 9 it can be seen that this limit is very close to the actual performance of the distributed allocation algorithm, confirming its correct operation. As the capacity of physical nodes increases, the actual performance diverges from the upper limit, as the finite capacity of links starts to impact the allocation process.

As the number of virtual network nodes increases, fewer virtual networks can be successfully allocated due to the faster resource depletion in physical nodes, as confirmed by the upper limit just discussed. In terms of the number of slots required by the virtual networks, a significant difference in the number of allocated virtual networks is not observed in this study.

Finally, in terms of topology, as buses require one link less per virtual network request, the number of established bus-topology virtual networks is higher than ring-topology ones. The difference between both topologies is higher at high node capacities, where the rejection of a new virtual network is influenced by the link utilisation rather than enough node capacity.

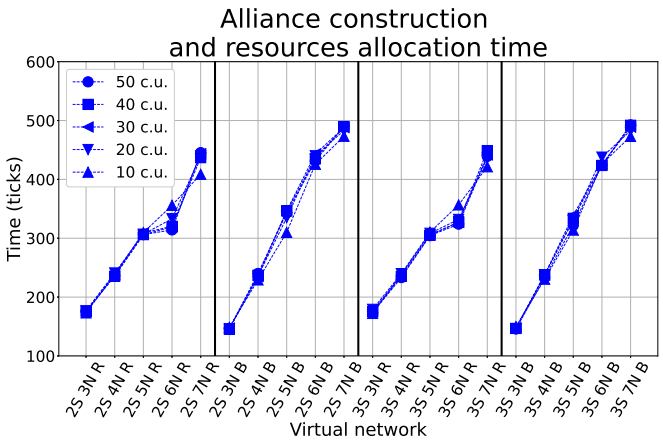


Fig. 10. Virtual network establishment time

A second metric studied was the time required to establish a virtual network, as it impacts the service offered to the customers. The vertical axis of the graph in Figure 10 corresponds to the average number of clock ticks required to establish a functional alliance. The horizontal axis corresponds to the different virtual networks studied, following the nomenclature of the previous figure.

A linear increase of time with the number of physical nodes can be observed. This is consistent with the protocol operation: During the virtual request processing, the number of messages sent by each agent is O(1). As a result, the total number of messages exchanged by the protocol is O($|NP|$), that is, linear with the number of physical nodes. In the studied cases, the maximum time to establish a virtual network is about 500 clock ticks. This number was obtained by dividing the time the light takes to travel a given distance in fiber by the number of time units elapsed in Netlogo to simulate light propagation through the same distance. This time is determined by the message exchange process only, not by the local computation performed by the agents. Whether the total time is faster than the time required by the centralised approach it is difficult to say. The centralised approach performs minimum message exchange (from the majordomo to the agents to start the allocation process), and from that point of view is certainly faster than the distributed algorithm that performs several rounds of message exchanges. However, the computational complexity of the centralised is higher than that of the distributed protocol. Whether this establishment time in the order of 0.1 seconds achieved by the distributed approach is fast enough depends on the dynamic of the virtual network arrival process. Due to the emergence of this type of service, no studies regarding this aspect have been published yet.

Unlike the number of virtual networks, the capacity of physical nodes does not significantly impact the time required to establish a virtual network. As a result, all curves are very similar. This behaviour is expected since the only effect of the physical node capacity occurs at the beginning of the algorithm, when the node eventually discards itself from the resource allocation process if it does not have sufficient capacity. After that decision, the node's capacity does not affect the algorithm's execution time.

In terms of the number of virtual network nodes, the higher this number, the longer it takes to establish a virtual network since more interactions between physical nodes are required to complete the alliance. Regarding the number of slots required by the virtual networks, a significant impact on the time required to establish a virtual network is not observed since checking the status of 2 or 3 slots in a link requires the same number of operations.

Finally, in terms of the topology of the virtual network, as the number of nodes increases, the bus topology takes slightly longer than the ring topology to be successfully established (10% longer in the case of 7-node virtual topologies). This result is unexpected, as the impact of the 1-link difference between both topologies should not be significant in terms of time (virtual links are established in a parallel manner). To find out the reason behind this behaviour, we studied the number of messages sent during the algorithm's execution. More messages will lead to longer allocation times.

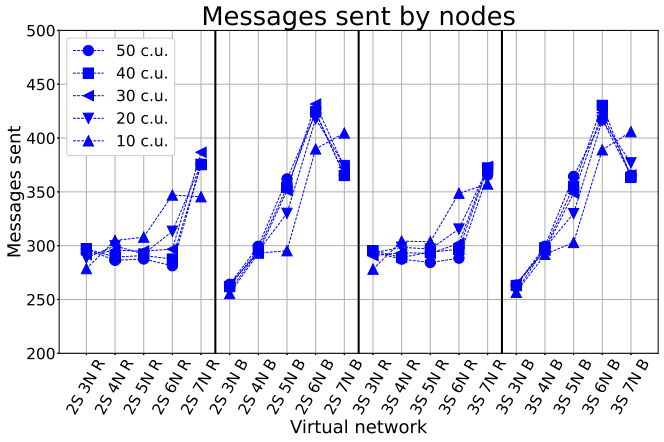


Fig. 11. Messages sent by nodes

Figure 11 shows the average number of messages sent by the physical nodes during the allocation of a virtual network. The horizontal axis corresponds to the different virtual networks studied. Each name of the horizontal axis follows the same nomenclature used in previous figures.

The total number of messages in the studied cases never exceeds a few hundred, and it grows with the number of virtual nodes, as expected: more virtual nodes require more synchronisation messages to enlarge the alliances.

In terms of virtual topologies, in line with the behaviour observed in Figure 10, as the number of physical nodes increases, the bus topologies exhibit a slightly higher number of messages

than ring topologies. After observing the visual behaviour of the allocation process in Netlogo, we can report that virtual networks with bus topologies generate more nodes in the state "Proposing alliance" that receive a [negative 2] response than ring topologies for several virtual nodes higher than 4. As a result, more messages are exchanged, and more time is taken to enlarge the alliance. This situation depends on the particular physical and virtual topologies and the way the virtual links are selected. Further research should study such behaviour in greater detail.

6. CONCLUSIONS

In this paper, a multi-agent distributed protocol for virtual network allocation over elastic optical networks was proposed for the first time and evaluated employing an agent-based simulation.

Due to the distributed nature of the protocol, its feasibility was proved by implementing it in an agent-based simulator. The protocol was programmed in Netlogo, where each node of the physical network corresponds to an agent. The protocol was evaluated in incremental traffic scenarios, using the NSFNet as the physical elastic optical network. Physical nodes were simulated with 10, 20, 30, 40, and 50 capacity units. Virtual networks with ring and bus topologies, with several virtual nodes ranging from 3 to 7 nodes and virtual links requiring 2 or 3 slots were used to study the algorithm.

Simulation results showed that the algorithm effectively allocates virtual networks, achieving a performance very close to the upper limit in terms of the number of virtual networks allocated when the finite capacity of links does not significantly affect the algorithm performance. As expected, the number of virtual networks successfully allocated increased with the network capacity and decreased with the number of virtual nodes. Regarding the time required to establish the virtual networks, it increased with the number of virtual nodes, and it was slightly higher in buses than rings for networks with 5 or more virtual nodes. In the worst case, the time was in the order of 0.1 seconds.

Future research should extend this work along the following lines of research: to evaluate the impact of the physical topology on the performance of the algorithm (including regular and mesh topologies) as well as the impact of the connectivity of the virtual networks (maintaining the number of nodes and increasing the number of virtual links from a bus to a clique); to explore different fitness functions; to evaluate the impact of fully dynamic traffic on the performance of the resource allocation process; to improve ways of assigning virtual nodes to physical nodes; extending the distributed protocol to include routes different from the shortest path to establish the virtual links as well as the effect of different modulation formats on the optical reach of routes; and studying the impact of using an explicit control mechanism instead of a timeout-based scheme to determine the rejection of a request.

FUNDING

Agencia Nacional de Investigación y Desarrollo (FONDECYT Iniciación 11220650, FONDECYT Iniciación 11190710, FONDECYT Iniciación 11201024, Doctorado Nacional/2021-21211075, Doctorado Nacional/2020-21200588, FOVI210082); Universidad Técnica Federico Santa María (PI_LII_2020_74, PIIC 018/2022, PIIC 007/2022); Pontificia Universidad Católica de Valparaíso (DI-PUCV 39.437/2020 and 039.382/2021).

REFERENCES

1. S. S. Manvi and G. Krishna Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," J. Netw. Comput. Appl. **41**, 424 – 440 (2014).
2. M. Gharbaoui, B. Martini, and P. Castoldi, "Anycast-based optimizations for inter-data-center interconnections [invited]," Opt. Commun. Networking, IEEE/OSA J. **4**, B168–B178 (2012).
3. O. Michel, E. Keller, and F. M. V. Ramos, "Network defragmentation in virtualized data centers," in *2019 Sixth International Conference on Software Defined Systems (SDS)*, (2019), pp. 17–24.
4. H. Cao, S. Wu, G. S. Aujla, Q. Wang, L. Yang, and H. Zhu, "Dynamic embedding and quality of service-driven adjustment for cloud networks," IEEE Transactions on Ind. Informatics **16**, 1406–1416 (2020).
5. M. Zangiabady, A. Garcia-Robledo, C. Aguilar-Fuster, and J. Rubio-Loyola, "A holistic framework for virtual network migration to enhance embedding ratios in network virtualization environments," J. Netw. Syst. Manag. **28**, 502–552 (2020).
6. Y. Zong, C. Feng, Y. Guan, Y. Liu, and L. Guo, "Virtual network embedding for multi-domain heterogeneous converged optical networks: Issues and challenges," Sensors. **20** (2020).
7. H. Halabian, "Distributed resource allocation optimization in 5g virtualized networks," IEEE J. on Sel. Areas Commun. **37**, 627–642 (2019).
8. "White paper: Where network virtualization fits into data center initiatives," Tech. Rep. 16VM066, VMware, Inc, 3401 Hillview Avenue Palo Alto CA 94304 USA (2016).
9. B. Mukherjee, I. Tomkos, M. Tornatore, P. Winzer, and Y. Zhao, *Handbook of Optical Networks* (Springer, 2020).
10. T. Portela, M. E. Monteiro, J. R. A. Cavalcante, J. Celestino Jr, and A. Patel, "An extended software defined optical networks slicing architecture," Comput. Standards & Interfaces **70**, 103428 (2020).
11. J. Zhang, B. Mukherjee, J. Zhang, and Y. Zhao, "Dynamic virtual network embedding scheme based on network element slicing for elastic optical networks," in *39th European Conference and Exhibition on Optical Communication (ECOC 2013)*, (2013), pp. 1–3.
12. W. Fan, F. Xiao, X. Chen, L. Cui, and S. Yu, "Efficient virtual network embedding of cloud-based data center networks into optical networks," IEEE Transactions on Parallel Distributed Syst. **32**, 2793–2808 (2021).
13. Y. Wang, Z. McNulty, and H. Nguyen, "Network virtualization in spectrum sliced elastic optical path networks," J. Light. Technol. **35**, 1962–1970 (2017).
14. O. Gerstel, M. Jinno, A. Lord, and S. J. B. Yoo, "Elastic optical networking: a new dawn for the optical layer?" IEEE Commun. Mag. **50**, s12–s20 (2012).
15. P. Layec, A. Morea, F. Vacondio, O. Rival, and J. Antona, "Elastic optical networks: The global evolution to software configurable optical networks," Bell Labs Tech. J. **18**, 133–151 (2013).
16. B. Chatterjee and E. Oki, *Elastic Optical Networks: Fundamentals, Design, Control, and Management* (CRC Press, 2020).
17. D. Bórquez-Paredes, A. Beghelli, and A. Leiva, "Network virtualization over elastic optical networks: A survey of allocation algorithms," in *Optical Fiber and Wireless Communications*, R. Roka, ed. (IntechOpen, Rijeka, 2017), chap. 2.
18. A. Rezaee, O. Akbari Sheikhabad, and L. Beygi, "Quality of transmission-aware control plane performance analysis for elastic optical networks," Comput. Networks **187**, 107755 (2021).
19. H. Cao, J. Du, H. Zhao, D. X. Luo, N. Kumar, L. Yang, and F. R. Yu, "Towards tailored resource allocation of slices in 6g networks with softwarization and virtualization," IEEE Internet Things J. pp. 1–1 (2021).
20. M. Siqueira, J. Oliveira, G. Curiel, A. Hirata, F. van't Hooft, M. Nascimento, J. Oliveira, and C. E. Rothenberg, "An optical sdn controller for transport network virtualization and autonomic operation," in *2013 IEEE Globecom Workshops (GC Wkshps)*, (2013), pp. 1198–1203.
21. D. King, A. Farrel, E. Nishida King, R. Casellas, L. Velasco, R. Nejabati, and A. Lord, "The dichotomy of distributed and centralized control: Metro-haul, when control planes collide for 5g networks," Opt. Switch. Netw. **33**, 49–55 (2019).
22. R. K. Jana, A. Mitra, A. Pradhan, K. Grattan, A. Srivastava, B. Mukherjee, and A. Lord, "When is operation over $\mathrm{C}+\mathrm{L}$ bands more eco-

nomical than multifiber for capacity upgrade of an optical backbone network?" in *2020 European Conference on Optical Communications (ECOC),* (2020), pp. 1–4.
23. I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *2008 IEEE International Conference on Communications,* (2008), pp. 5634–5640.
24. M. T. Beck, A. Fischer, J. F. Botero, C. Linnhoff-Popien, and H. de Meer, "Distributed and scalable embedding of virtual networks," J. Netw. Comput. Appl. **56**, 124–136 (2015).
25. F. Esposito, D. Di Paola, and I. Matta, "On distributed virtual network embedding with guarantees," IEEE/ACM Transactions on Netw. **24**, 569–582 (2016).
26. K. T. Nguyen and C. Huang, "An intelligent parallel algorithm for online virtual network embedding," in *2019 International Conference on Computer, Information and Telecommunication Systems (CITS),* (2019), pp. 1–5.
27. A. Song, W.-N. Chen, T. Gu, H. Yuan, S. Kwong, and J. Zhang, "Distributed virtual network embedding system with historical archives and set-based particle swarm optimization," IEEE Transactions on Syst. Man, Cybern. Syst. **51**, 927–942 (2021).
28. S. F. Railsback and V. Grimm, *Agent-Based and Individual-Based Modeling: A Practical Introduction* (Princeton University Press, 2011).
29. "IEEE Standard for Ethernet," IEEE Std. 802.3-2008 (2012).
30. U. Wilensky, "Netlogo," http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL (1999).
31. H. Ballani, T. Karagiannis, A. Rowstron, and P. Costa, "Towards predictable datacenter networks," in *The ACM SIGCOMM Conference on Data Communication (SIGCOMM'11),* (2011).
32. C. Fuerst, S. Schmid, L. Suresh, and P. Costa, "Kraken: Online and elastic resource reservations for multi-tenant datacenters," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications,* (2016), pp. 1–9.
33. M. Rost, C. Fuerst, and S. Schmid, "Beyond the stars: Revisiting virtual cluster embeddings," SIGCOMM Comput. Commun. Rev. **45**, 12–18 (2015).
34. C. Guo, G. Lu, S. Yang, C. Kong, P. Sun, W. Wu, Y. Zhang, H. Wang, and G. Lv, "Secondnet: A data center network virtualization architecture with bandwidth guarantees," in *ACM CONEXT 2010,* (Association for Computing Machinery, Inc., 2010).
35. F. I. Calderón, A. Lozada, D. Bórquez-Paredes, R. Olivares, E. J. Davalos, G. Saavedra, N. Jara, and A. Leiva, "Ber-adaptive rmlsa algorithm for wide-area flexible optical networks," IEEE Access **8**, 128018–128031 (2020).
36. F. Falcón, G. España, and D. Bórquez-Paredes, "Flex net sim: A lightly manual," arXiv preprint arXiv:2105.02762 (2021).
37. D. Bórquez-Paredes, A. Beghelli, , R. Olivares, G. Saavedra, N. Jara, A. Lozada, A. Leiva, and P. Morales, "Agent-based distributed protocol and centralised algorithm," https://gitlab.com/IRO-Team/agents-optical-network-protocol (2022). Visited on 2022-07-04.