CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2013; **00**:1–19
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Tailoring the Network to the Problem: Topology Configuration in Hybrid EPS/OCS Interconnects

[1]Kostas Christodoulopoulos, [2]Kostas Katrinis, [1]Marco Ruffini, [1]Donal O'Mahony

1: CTVR and School of Computer Science and Statistics, Trinity College Dublin, Ireland,
2: IBM Research - Ireland

SUMMARY

We consider a hybrid Electronic Packet Switched (EPS) and Optical Circuit Switched (OCS) interconnection network for future High Performance Computing (HPC) and Datacenter (DC) systems. Given the logical task-to-task communication graph of an application, our objective is to cluster the logical parallel tasks to compute resources and configure the (re-configurable) optical part of the hybrid interconnect to efficiently serve application communication requirements. We formulate the Clustering and Topology-Configuration (CTC) problem in such a network, prove that it is NP-complete and provide an optimal algorithm to solve it based on an Integer Linear Programming (ILP) formulation. The ILP algorithm is used to optimally solve small-scale instances of the problem for the purpose of obtaining performance bounds. Aiming at large-scale, we also present a heuristic based on Simulated Annealing that trades-off performance for responsiveness. We measure the performance of a hybrid interconnect employing the proposed algorithm using real workloads, as well as extrapolated traffic, and compare it against application mapping on conventional fixed, electronic-only interconnects based on toroidal topologies. Copyright © 2013 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Reconfigurable interconnection networks, optical circuit switching, communication graph, topology-aware application mapping, clustering, topology configuration

1. INTRODUCTION

High Performance Computing (HPC) and Datacenter (DC) systems are being built out of ever-increasing numbers of processors, and their scale is expected to grow to the order of millions of cores towards Exascale [1]. To obtain high system efficiency, computation vs. communication performance needs to be balanced. Given the aggressive increase rate in compute density - thanks to increasing the number of cores per node and the growing deployment of acceleration units - it is of paramount importance to avoid having the interconnection network become the bottleneck [1]-[2]; instead, interconnect technologies and supporting system software need to grow in hand with the evolution in compute density to enable next generation HPC and DC systems.

Supercomputers typically employ direct networks with regular topologies, such as hypercubes and torus structures. For instance, Cray XE6, Fujitsu K and IBM BlueGene/Q all utilize 3- or higher-dimension torus topologies [3]-[5]. Such low degree regular topologies are adopted due to their efficient forwarding operations and good cost scaling. Still, these sparse topologies tend -for specific application classes- to complicate the mapping of the communication to the fixed physical connectivity [6]-[12], while the problem becomes more difficult for applications with irregular

*Correspondence to: K. Christodoulopoulos: email christok@tcd.ie, CTVR, Trinity College Dublin, Dublin 2, ireland

communication [13]. On the other hand, many HPC clusters and DCs use indirect networks (e.g. fat-trees [15]) that simplify the mapping of applications with arbitrary communication requirements to the underlying network. Though, this comes at super-linear cost scaling and thus raises concerns when considered at extreme scale, especially if the investment is not justified due to poor utilization [16]. To alleviate this, oversubscription [17] has been proposed as a means of controlling the capital and power-consumption costs. Still, this approach comes with the inflexibility of placing network capacity once and for all at design phase, hindering the adaptation of capacity to changing needs.

To close the gap between inflexible regular and costly tree-based interconnection networks, past work [18]-[23] has proposed building low-degree, reconfigurable interconnects that allow for on-demand bandwidth allocation wherever it is needed. Maintaining a low-degree interconnect reduces its contribution to the total capital expenditure of the system, while these designs have been shown to fit well with the communication patterns exhibited by specific classes of HPC applications [18] and DC workloads [21]-[23]. The proposed reconfigurable interconnect architectures are hybrids of two switching technologies, using both electronic packet switches (EPS) and optical circuit switches (OCS). The OCS part of the network is reconfigurable, typically implemented with commodity Micro Electro-Mechanical Systems (MEMS) optical switches that can be driven by commodity optical transceivers (note that optical transceivers are already deployed in datacenters for communication beyond the rack level). The OCS part of the network handles high-rate, long-lived point-to-point flows while the EPS part serves low-rate signaling traffic, short-lived flows and collectives (many-to-many communication). To distinguish between the hybrid EPS/OCS and the standard practise interconnects that are based only on EPS, we will refer to the second as an *EPS-only* or *electronic-only* system for the remaining of this paper.

In this paper, we focus on traditional HPC applications that exhibit *static logical task-to-task communication*. We call an application static if it follows specific communication patterns so that its logical communication graph at intermediate phases and the aggregated graph at the end of its execution have well defined and consistent structures irrespective of the application input. This definition is very close to the definition given in [18], where a static application is considered to have known communication pattern at compilation time. Note that the term *logical* implies that the task-to-task communication is influenced only by the application, and thus does not depend on machine or interconnect characteristics. The majority of HPC parallel applications that follow the MPI (Message Passing Interface) standard fall into this category, with one of the exceptions being the adaptive-mesh-refinement (AMR) MPI applications that dynamically change their processing/communication behaviour depending on the input. Work on identifying and classifying the task-to-task communication graphs of static HPC applications include [24] and [25]. Other types of applications apart from MPI can also exhibit a static communication pattern, e.g. a MapReduce job where after placing the Mapper and Reducer tasks these tasks communicate in a point-to-point or multipoint manner, or a stream computing application [26] where processing tasks form a graph and information flows over that in a point-to-point or multipoint manner.

Given an HPC system with an EPS-only interconnect, mapping the tasks of a parallel application onto the physical processors to allow efficient communication is one of the critical performance issues. This is usually referred to as topology-aware mapping (mapping being equivalent to the task-to-processor assignment and the topology referring to the static topology of the EPS-only underlying interconnect) and has been shown to be NP-complete [6]. The most efficient known approach [8] is to adapt the application source code to optimize communication against the specific static topology. Albeit efficient, this approach is cumbersome, and precludes the re-use across machines or applications. More practical alternatives [12]-[14] trade-off efficiency for being application-agnostic. Specialized system software maps task to processing elements (or aggregations of processing elements such as e.g. multiprocessor nodes, or racks). Typically, they take as input the static logical communication graph of an application and perform the task mapping taking into account - among others - the static underlying topology. The mapping that optimizes communication throughout application execution is computed once and is then reused during future repetitions of the application, since static applications exhibit similar communication patterns independent of the input.

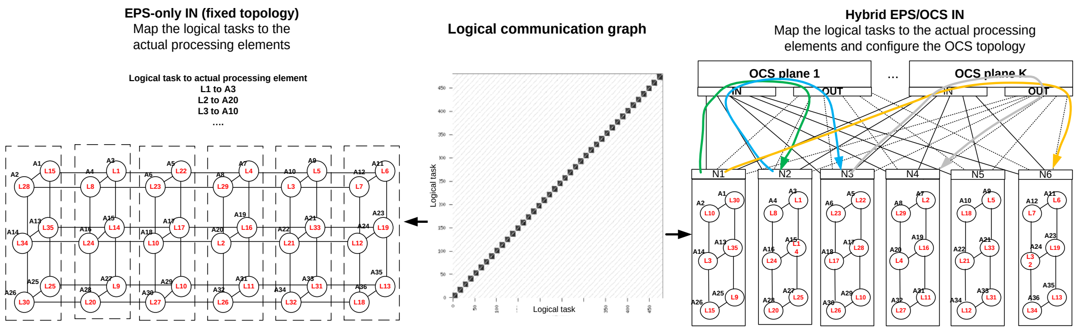
EPS-only IN (fixed topology)
Map the logical tasks to the actual processing elements
Logical task to actual processing element
L1 to A3
L2 to A20
L3 to A10
....
Logical communication graph
Logical task
Logical task
Hybrid EPS/OCS IN
Map the logical tasks to the actual processing elements and configure the OCS topology
OCS plane 1
IN
OUT
...
OCS plane K
IN
OUT
N1
N2
N3
N4
N5
N6

Figure 1. Topology-aware mapping on fixed electronic-only interconnect (left) and topology configuration in reconfigurable OCS interconnect (right).

Among the various advantages brought by reconfigurable interconnects [18]-[23], the ability to look at task mapping from a different angle remains unexplored: instead of performing a sophisticated topology-aware mapping as in [6]-[14], the tasks can be clustered (into aggregations of processing elements) and the topology of the (reconfigurable) OCS part of the network that interconnects these clusters can be configured for optimized communication and thus improved application performance. We call this the Clustering and Topology Configuration (CTC) problem. Fig. 1 contrasts the two different approaches.

In this paper, we assume that the logical task-to-task communication graph of a static application is identified at compilation time or the application is executed once to profile and capture its communication graph. Using this input, an optimized Clustering and the Topology Configuration (CTC) solution is computed and used to speedup subsequent executions. Note that we assume that the application will run on dedicated resources without being interfered with other applications. This is common practise in HPC systems but also in multi-tenant datacenters where clusters are allocated to running specific jobs. We assume that the tasks are clustered and assigned to processing elements and the OCS network is configured once, that is, it is configured at the outset of application execution and remains the same throughout its execution. Although the reconfigurable network could dynamically adapt its topology to more efficiently support the different application phases, or to optimize the network for more than a single application, this optimization is out of the scope of this paper and constitutes part of our future plans. The proposed static configuration approach has merits in its own, and it is equivalent to the topology-aware mapping of HPC application as studied in [6]-[14]. For the static application cases, our performance results show that even configuring the hybrid, reconfigurable EPS/OCS interconnect once before the application execution can improve its communication performance. The advantage of the adopted reconfigurable architecture in this case comes from the ability of employing different optimized configurations for different applications.

We prove that the CTC problem in a hybrid EPS/OCS interconnect is NP-complete and develop an Integer Linear Programming (ILP) formulation that is used to optimally solve small-scale problem instances for the purpose of obtaining performance bounds. Aiming at large-scale, we also present a heuristic algorithm based on Simulated Annealing that trades-off performance for responsiveness. We use hop-bytes [11]-[14] as the performance metric and examine the performance for various architectural choices of the hybrid target EPS/OCS interconnect, the level at which optical interconnection occurs and the number of optical planes available. We evaluate the effectiveness of the proposed CTC heuristic algorithm in small problem instances by comparing it to the optimal ILP algorithm. For the purpose of realistic evaluation, we profile several static HPC kernels using IPM [27] and derive their logical task-to-task communication graphs. Then we compare the performance of the CTC heuristic algorithm against application mapping on conventional fixed, electronic-only interconnection topologies. Our results indicate that the hybrid interconnect can exhibit better average performance than electronic-only toroidal interconnects of higher connectivity degrees. By evaluating networks with the same connectivity degree, we observed improvements as high as 42% in hop-bytes. Subject to the relative costs of the electronic and optical ports, these performance improvements can come at comparable costs, while the data rate agnostic and transparent nature of optical technology ensures better upgradability and lower power consumption for the hybrid interconnect. This paper extends our work in [28] by developing an ILP formulation to optimally solve the CTC problem and estimating the optimality performance of the proposed heuristic algorithm as opposed to the ILP.

The rest of this paper is organized as follows. In Section 2 we report on previous work. In Section 3 we describe the architecture of the hybrid EPS/OCS system that we study. In Section 4 we formally define the Clustering and Topology-Configuration (CTC) problem in a hybrid interconnect and present an optimal ILP formulation and a heuristic algorithm to solve it. Our performance results are then presented in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

Many researchers have focused on topology-aware mapping as a way to optimize the execution of applications [6]-[14]. In fixed electronic-only interconnection networks the problem is defined with the logical communication graph of the application as input and the goal is to assign the tasks to the actual processing elements of the system so as to optimize the communication for the underlying topology. The authors of [12] examine the problem of mapping 2D communication patterns to a 2D and 3D torus interconnect. In a similar manner, [9] also examines the problem of mapping a variety of structured patterns on a BlueGene 3D torus interconnect. Apart from mapping well structured communication graphs, mapping of irregular graphs is examined in [12]. A Monte-Carlo technique to solve the mapping problem was proposed in [10]. Some older works on the topic include [6]-[7]. Considering a slightly different system setting, [14] proposes a hierarchical mapping process. In this setting, a high bisection bandwidth network interconnects the processing elements within an aggregation/cluster (first level of the interconnect hierarchy) and a slower network interconnects the different clusters together (second level of the interconnect hierarchy). In one embodiment of this, the clusters can be the boards that form the backplane for connecting many multi-core sockets together. In this hierarchical setting, the intra-cluster communication at the first level is considered negligible and the goal is to optimize the inter-cluster communication at the second level. To perform the mapping in this hierarchical system, [14] proposes an algorithm that (a) clusters the tasks and then (b) maps them to the actual resources, so as to optimize inter-cluster communication. The topology-aware mapping problem as examined in [6]-[14] is a generalization of the graph embedding problem which is known to be NP-complete. The graph to be embedded is the logical communication graph of the application and the surface that this is embedded in is the underlying electronic-only network. Thus, the majority of the above works provide heuristic algorithms to solve the related problem.

In contrast to the fixed electronic-only interconnect, in this paper we consider a reconfigurable interconnect, and so instead of performing a sophisticated mapping we configure the topology of the interconnect to serve the traffic in an efficient way. We focus on a hybrid Electronic Packet Switched (EPS) and Optical Circuit Switched (OCS) reconfigurable interconnection network [18]. A number of different variations of this architecture have been also proposed for DCs [21]-[23]. In these hybrid systems the EPS network connects all the processing elements of the system together. The OCS network is connected to certain points of the EPS network, e.g. at the top of rack (ToR) EPS switches. Directly connecting the processing elements of the system with the OCS network is not considered efficient, due to scalability issues and the slow reconfiguration times of the OCS switches. For instance, in [21]-[23], the traffic is aggregated, so that it changes more slowly, is not bursty and requires higher capacity, characteristics that best suit the OCS network.

Some of the basic differentiation points among the architectures proposed in [18]-[23] are: the aggregation level at which the OCS network is connected, the number and bandwidth of optical ports and the consideration of single or multi-hop connections. In particular, in the Helios architecture proposed in [21], the optical network interconnects the pods (a pod is a set of racks), while in c-Through [22] and in OSA [23] it interconnects the racks of the system. Only single-hop transmissions over the OCS network are considered in [21] and [22], while [23] considers multi-hop transmissions, although the proposed heuristic optimizes only the single-hop transmissions. Note that the consideration of multi-hop transmissions makes the related optimization problem computation intractable as we prove in this work.

In this paper we adopt an interconnect architecture that is similar to [18]-[22], in the sense that it is based on a commodity EPS network to which commodity optical switches are added. So the adopted architecture differs from the complex architecture of [23] which introduces active (wavelength selective switches) and passive (circulators) optical components and multiwavelength transceivers. Compared to previous approaches, we define a general system model without specifying explicitly the point at which the optical network is interconnected and the number of optical planes and ports. So our proposed methods are general and applicable to a variety of system architectures. Compared to [21] and [22] we consider the employment of multi-plane optical networks, and the support of multi-hop transmissions. Another difference is that in this paper we focus on what we call static configuration. In particular, we consider applications for which we assume that we know their communication graph beforehand. We assume that the OCS network is configured once for each application, that is, it is configured at the beginning of the application and remains the same throughout its execution. Although the considered reconfigurable network could dynamically adapt its topology during the application execution, we will not study this here. Note that this work does not cancel the potential of dynamic reconfiguration that can bring even more significant benefits. Our goal is to define a generalized system model, understand the complexity of optimizing the communication of a single static instance, and propose efficient algorithms to solve it.

A next step would be to consider the dynamic reconfiguration of the system which can be viewed as a sequence of static configuration instances. Note that there are a number of delays that are introduced when reconfiguring the hybrid network, as opposed to electronic-only networks that switch at packet-level granularity. First of all, the reconfiguration time of MEMS switches is relatively high (tens of msec), but there are more delays introduced by the need to reconfigure the electronic part of the hybrid network that faces the OCS network. We measured that the whole reconfiguration process can take up to hundreds of msec on a system prototype we are building (we plan to report on this in a following publication). This indicates that reconfiguration should be done not faster than tens of secs to achieve acceptable network efficiency. We plan to extend our proposed algorithm to consider the previous state of the network so as to reduce the number of reconfigured cross-connections, as a way to minimize the control plane delays. However, static configuration that we study here is also interesting in its own merit. Specifically, we show that even a static configuration of the studied hybrid interconnect can benefit existing applications when compared to the equivalent topology-aware mapping on electronic-only interconnects as studied in [6]-[14].

3. SYSTEM MODEL

We consider a generic multi-rack system architecture in which the processing is performed with multi-core processor chips. A given number of chips is contained within a (compute) node. In turn, a set of compute nodes comprises a rack. We assume an interconnection network adhering to a hybrid architecture comprising both an Electronic Packet Switched (EPS) and an Optical Circuit Switched (OCS) network [18]. The OCS network is typically implemented with one or more Micro Electro-Mechanical Systems (MEMS) optical switches (crossbars). Other technologies, e.g. using piezoelectric actuators [36], can be also used, as long as they support crossbar operation. MEMS switches are layer-0 (PHY) switches that establish a point-to-point optical connection by reflecting the light beam from an input to an output port. The signal is switched transparently without performing any processing on switched data. MEMS switches can operate with both multi-mode and single-mode fibers, the former option imposing limitations on the port count of the switch. Since these switches are agnostic to protocol and modulation format, any type of optical transceiver can be used with them. Note that optical transceiver and fibers are already widely used in datacenters for performing communication above the rack level. Typically, in datacenters multi-mode fiber short-reach transceivers are used, which have a reach of up to 400 m and consume less than 1W power. Long-reach transceivers that operate over single-mode fibers can reach up to 10 km with slightly higher consumption, but are mainly used for telecommunication applications. So, optical switches are compatible with technologies and equipment that are considered as a standard practise in building current generation datacenters.

Optical switches exhibit circuit setup times that are typically in the order of tens of msec and thus would result in prohibitively high per-packet switching overhead, should they be operated as packet switches (e.g. at 10Gbps the duration of a 1500-byte packet is 0.1 μsec, at least four orders of magnitude shorter compared to the reconfiguration time). Therefore, it makes sense to use the OCS for high-rate, long-lived, and point-to-point flows. Flows at core- or chip-level currently do not have such characteristics. Moreover, since optical switches support solely point-to-point communication, creating a chip-level or a node-level network would require a high-degree topology, which is prohibitive in terms of scalability (available MEMS switches reach up to 320 ports, high-degree networks can be built by interconnected many switches in a Clos topology, but the cost of such an interconnect would be very high). Therefore, OCS is applied in our system to aggregated traffic at a high level of the system hierarchy at what we call the *optical aggregation level*.

Given current cores/node figures, setting the optical aggregation at pod or rack level maximizes the utilization of optical circuits and reduces the frequency of reconfiguration of OCS-switches. Still, as compute density keeps increasing, so will the number of compute cores and the bandwidth per node [1], potentially to a level where placing the optical aggregation at the node level will start being justified (e.g. a few hundreds of cores on a node could justify such a choice). However, as stated above, there are scalability issues related to the optical switches that have to be solved in order for such an OCS network to be realized. To allow our work to capture this trend and thus be future-proof, we apply an abstraction to the assumed system model that enables us also to carry out parametric studies.

Specifically, we create a general system model that represents a hierarchical interconnect. We assume that T processing elements (cores) are *clustered* together to form groups that we refer to as *Logical Clusters* (*LCs*). The processing elements comprising a LC are interconnected via an EPS network (to be referred to as the first-level of EPS network). The first-level of EPS serves both intra-LC communication (communication among the T processing elements comprising the LC), as well as aggregates traffic destined to distant LCs at its edge (this edge comprises the optical aggregation level). At the optical aggregation level M EPS (bidirectional) ports are connected to the second-level of the EPS network (EPS-core) and also $P \cdot K$ (bidirectional) ports are connected to the OCS network. In particular, we assume the use of K optical crossbar switches and P ports connected to each of these switches from each LC. So from the OCS network perspective, the network end-points are the LCs and T processing elements are accessible at each such endpoint.

While the above abstraction enables us to evaluate our approach for different optical aggregation levels, it still allows a straightforward mapping of our approach to real systems. To showcase this, we assume an HPC cluster where the optical aggregation point can be e.g. a multi-port Ethernet NIC (node-level optical aggregation) or an Ethernet rack or pod switch (rack or pod level optical aggregation). The processing elements inside the LCs (nodes, racks or pods, respectively), are interconnected via an EPS network (first level) while electro-optical transceivers attached to the LC network edge (NICs or switches' ports) are used to carry packets to/from the OCS network and the second level of the EPS network.

The hierarchical interconnect under study has an EPS network at the first level and two parallel networks at the second level. So, at the optical aggregation level the traffic towards distant LCs can be served by two networks: the EPS-core or the OCS network. The OCS network handles persistent point-to-point, high-rate inter-LC flows, while the EPS-core network handles lower bandwidth and collective communications, as well as bursty flows. Note that interconnects with two levels of hierarchy are also found in fixed topology EPS-only systems [12] and this architecture is followed in all related works for hybrid EPS/OCS interconnects [18]-[23].

Let N denote the number of LCs of the system, each one comprising of T processing elements (cores). In the hybrid interconnect architecture that we adopt, each LC is connected with P ports to

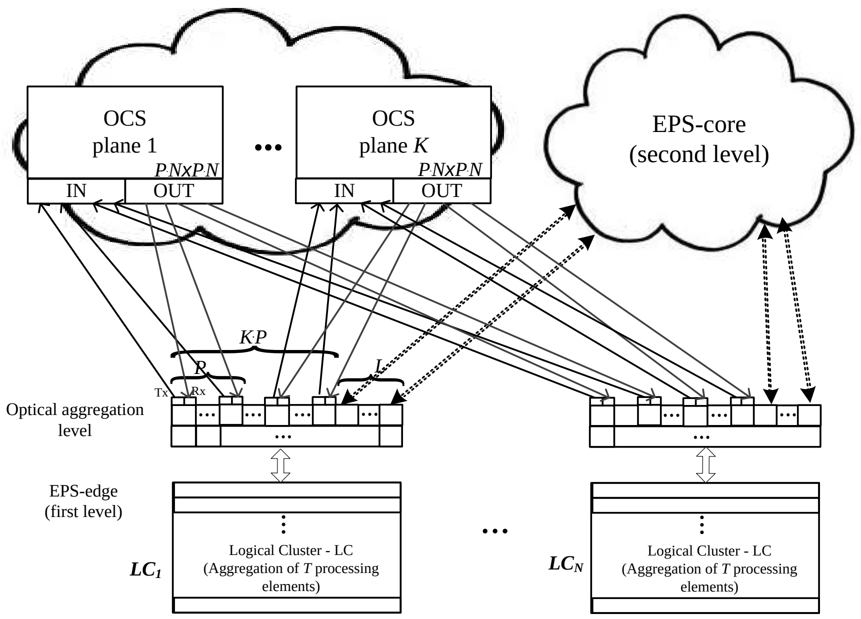

Figure 2. Reference hybrid EPS/OCS interconnect consisting of N logical clusters (LCs). At the optical aggregation level, M and $K \cdot P$ bidirectional EPS ports are connected to the EPS-core and the OCS network, respectively. In particular, P ports from each LC are connected to each of the K optical switches (optical parallel planes). At each of the EPS ports accessing the OCS network, the transmit (receive) side of the transceiver is connected to an input (output) port of the related OCS switch.

each of K parallel OCS planes at the optical aggregation level. Fig. 2 presents the general model of the system. Since, each parallel OCS plane is implemented with a $(P \cdot N)$x$(P \cdot N)$ crossbar switch, K crossbar switches of radix $(P \cdot N)$x$(P \cdot N)$ are required in total. Note that a large OCS crossbar switch can be partitioned to create smaller crossbar switches, and so the total number of switches required can be less than K. Currently 320x320 MEMS crossbar switches are available from a number of vendors and there are prototypes with more than 1024 in/out ports. Even larger switches with thousands of ports can be designed by connecting switches in a multistage fashion. Note that our goal is to create a hybrid EPS/OCS network with K, P<<N to keep the cost as low as possible and enable massive scale out.

We assume that all network devices connect via a low-rate network (management network) to a dedicated server, where a network controller is running. The main role of the controller is to periodically or upon application request configure the optical part of the network in a manner that benefits the applications. The network controller runs an algorithm such as the one described in the following section to calculate the optical topology and then the controller configures the optical switch(es) and the edges of the first-level EPS network (optical aggregation level) to forward the traffic over the OCS and the EPS-core networks. The key idea to split the traffic without affecting applications execution is to apply appropriate switching/routing rules (decisions taken by the algorithm) at the EPS-edge. To support this operation, Vlan-based routing similar to [29] or Software Defined Networking solutions, e.g. Openflow, can be used.

In this paper we focus mainly on the algorithm to configure the optical part of the hybrid interconnect. The methods we propose are general and can be used in many different settings, irrespective of the optical aggregation level and the number of parallel OCS planes and ports. The methods take as input the logical task-to-task communication graph of an application, given e.g. in the form of a traffic matrix, and a specific interconnect architecture as defined by the related K, P, and T parameters. We assume that the rest of the traffic that is not point-to-point (e.g. collectives) is routed over the EPS-core part of the hybrid interconnect. The goal of the proposed methods is to serve the task-to-task transmissions in an efficient way. To do so, we cluster tasks (to form LCs) and map these to processing resources and also derive the configuration of the OCS network to optimize the communication between the formed LCs. The clustering process divides the communication into inter- and intra-LCs communication, similar to the hierarchical mapping problem examined in [14]. Intra-LC communication that is served by the first-level EPS is considered to be cheap and can be efficiently served using established network technologies, while inter-LC communication, which is routed over the OCS network, is the communication that we target to optimize.

Depending on the inter-LC traffic, the communication can be mapped to the OCS network so as to serve all connections over single-hop transmissions or we have to resort to multi-hop transmissions. Multi-hop refers to communication over the OCS that involves electronic processing of the packets at the EPS-edge, beyond the source/destination LCs, that is at intermediate LC hops. So, intermediate LC hops, apart from sourcing traffic to the OCS network, are also relaying traffic, in a similar manner that direct electronic-only interconnects, such as torus, hypercubes, etc, do. Multi-hop routing increases the effective bandwidth between LCs at the expense of increased latency as well as increased congestion and higher average network load. Since the OCS network supports only point-to-point communication, if we don't use multi-hop connections we can connect only a limited number of the network end-points which are the LCs. Depending on the application, this could have a small or a substantial effect on its performance. Assuming an application where many end-points communicate (inter-LC traffic has high connectivity degree), using multi-hop connections, would substantially improve the application performance. On the other hand, an application that has a sparse inter-LC traffic matrix that could be served over single-hop connections would not obtain any benefit. Finally, supporting multi-hop optical connections has an additional advantage, since it can reduce the need for frequent reconfiguration of the optical switch(es).

4. CLUSTERING AND TOPOLOGY-CONFIGURATION PROBLEM

In this section we formulate the Clustering and Topology-Configuration (CTC) problem in the hybrid EPS/OCS interconnect under study, derive an optimal - albeit exponential - solution and provide for an efficient heuristic solution to the problem.

We start with a parallel application that utilizes *Z* tasks. A typical allocation would consist of assigning each task to a processing element corresponding to a core, although other assignments at the thread or chip level are also applicable. We assume that we are given the logical task-to-task communication graph of the application in the form of a traffic matrix ***D*** of size *Z*x*Z*. Element $D_{nm}(1 \leq n, m \leq Z)$ corresponds to the task-to-task communication volume (in bytes) exchanged between tasks *n* and *m* throughout the time that ***D*** refers to; thus ***D*** has a zero diagonal. The remaining transmissions that are not point-to-point (e.g. collectives) are routed over the EPS-core part of the hybrid interconnect and are not considered here. Tasks are first clustered into logical clusters (LCs). In particular, we assume that *T* tasks are grouped together and mapped/assigned to a LC (one of the LCs may contain less than *T* elements, if Z mod $T > 0$). Grouping *T* tasks together stems from the architecture where *T* cores are reachable at each end-point of the optical network. The number of LCs *N* after clustering the *Z* tasks is given by $N = \lceil Z/T \rceil$.

By forming the LCs we transform traffic matrix ***D*** into a new traffic matrix ***d*** of size *N*x*N*, with each element $d^{ij}(1 \leq i, j \leq N)$ corresponding to the point-to-point communication volume (in bytes) between LC_i and LC_j, i.e. ***d*** corresponds to the inter-LC communication graph. Note that unlike ***D*** being part of the problem input, ***d*** depends on the clustering decision and as such is an intermediate output. Inter-LC traffic, as described by traffic matrix ***d***, is served by the reconfigurable OCS network. The LCs are mapped to the physical compute resources, by putting tasks that belong to the same LC together in the same compute aggregation (nodes or racks) without considering the position of the aggregation in the interconnect topology, since the OCS network will be configured around these aggregations. As described in Section 3, the OCS network we adopt consists of *K* parallel OCS planes and each LC/compute aggregation is connected with *P* ports to each plane. Each of the *K* parallel OCS planes can be configured to connect any LC to any LC, with the constraint that at most *P* connections initiate from or terminate at each LC on each plane. There are two different versions of the problem: assuming unidirectional or bidirectional connections over the OCS network. This is mainly related to the type/configuration of the switches at the optical aggregation level. Given that the unidirectional case is more generic and that commodity EPS switches can support this operating mode, we will focus on unidirectional connections, although our algorithms can be applied with minor changes to bidirectional connections. Note that if traffic matrix $\boldsymbol{d}$ includes both directions of communication for a LC pair, the solution will include connections for both directions, although they may utilize different (multi-hop) paths.

The OCS network is represented by a multi-layered graph $\boldsymbol{G} = (\boldsymbol{V_k}, \boldsymbol{E_k})$, $k = 1, ..., K$. $\boldsymbol{V}$ is the set of vertices that correspond to the logical clusters (LCs), $|\boldsymbol{V}| = N$. Vertices $\boldsymbol{V}$ are replicated K times in the layered graph $\boldsymbol{G}$, once for each of the K parallel OCS planes, to create $\boldsymbol{V_k}$. At each plane (layer) k, the edges $\boldsymbol{E_k}$ form a fully connected graph that represents the set of candidate links that connect every pair of $\boldsymbol{V_k}$. Actually, the edges in $\boldsymbol{E_k}$ do not correspond to physical links but are the candidate links that can be established in the OCS network. In this context, we will say that we establish an optical link (k, i, j) when we configure the kth-plane of the OCS network to establish a circuit connection between LC_i and LC_j. Since we are considering unidirectional connections (directed edges), the order of i and j is important. The number of links that can be established in each plane is constrained by the number P of ports per plane. We let $\boldsymbol{L_k} \subseteq \boldsymbol{E_k}$ be the set of links that are chosen to be established on the kth-plane, and $\boldsymbol{G'} = (\boldsymbol{V_k}, \boldsymbol{L_k})$ the related graph derived. In this formal notation, $\boldsymbol{L_k}$ constitutes the chosen crossconnection instance of the kth optical switch and graph $\boldsymbol{G'}$ corresponds to the topology of the configured OCS network.

The Clustering and Topology-Configuration (CTC) problem is defined as follows. We are given the traffic matrix $\boldsymbol{D}$, the number T of tasks per LC, the number K of parallel OCS planes and the number P of ports per LC per plane, and we seek to identify a “good” clustering of tasks to LCs (i.e. find the $\boldsymbol{d}$ matrix), and a “good” configuration of the OCS network that interconnects the LCs (i.e. identify the set $\boldsymbol{L_k}, k = 1, ..., K$ of optical links to be established). The goodness of a clustering and topology-configuration solution is measured by its ability to minimize a communication objective, namely average hop-bytes. The average hop-bytes metric is defined as the sum of path lengths taken by messages, weighted by the respective message sizes [11]-[14]. The hop-bytes metric measures the load that crosses the network and thus captures approximately the congestion and the latency to be experienced. In early works in the fields of graph embedding and VLSI design, emphasis was placed on the maximum dilation metric, which is the longest path (i.e. the longest wire in a circuit). [12] argues that reducing the longest path traversed by any message (maximum dilation) is not as critical as reducing the average hops across all message sizes, as captured by the hop-bytes metric. Also compared to bisection bandwidth that is calculated for a specific traffic pattern, the hop-bytes metric takes into account the actual traffic of the application and suits better the particularities of the reconfigurable interconnect architecture that we adopt here. Note that our algorithmic formulations are general and can be used to optimize any communication performance metric, even the maximum dilation or bisection bandwidth, as long as they can be formulated as linear expressions of the flow variables that describe the problem.

If the optical aggregation level is placed directly at the processing elements, then clustering becomes obsolete. This is a special case of the CTC problem with T=1 and thus $\boldsymbol{d}$= $\boldsymbol{D}$. In this case, the problem is reduced to the topology-configuration problem with multi-hop transmission, which we prove in the Appendix to be NP-complete by a reduction to the circular arrangement problem [30]. This case covers also the version of the problem where we are given directly the traffic matrix $\boldsymbol{d}$ capturing the traffic that will be routed over the OCS network. A use case for this is the dynamic execution of an application in which the traffic is monitored at the optical aggregation level and the OCS network is configured to efficiently serve this traffic, e.g. as done in [21]. Note that the support of multi-hop transmissions makes the problem intractable, otherwise the topology configuration problem for single-hop connections can be solved with variations of the perfect matching algorithm that is of polynomial complexity (this is the actual approach followed in [21]-[23]). Since the general problem described above with $T \geq 1$ includes as a special case an NP-complete problem, the general problem is also NP-complete.

The optimal solution of CTC is bound to give at least as good performance as the application mapping on hierarchical electronic-only fixed interconnects [14], as long as the connectivity degree of the hybrid network is not less than that of the fixed interconnect. For example, assuming that the optimal topology to serve the traffic is a 3D torus, the optimal CTC solution would be to configure the OCS network to form a 3D torus, assuming e.g. $K = 6$ and $P = 1$.

Lastly, note that in the above problem definition we consider that the communication is described in volume (the traffic matrix $\boldsymbol{D}$ is expressed in bytes), since we assume that we don't have detailed temporal information about the application's execution. However, this problem definition can also capture a similar problem setting where rates are used instead of volumes, i.e. by having the traffic matrix representing average or peak communication rate (bps) between the tasks instead of volume (bytes). This problem is a variation of the problem described above, with the addition of having to consider capacity constraints over the established optical links. The capacity constraints stem from the type/rate of transceivers that are utilized and not from the type of OCS network, since MEMS OCS switches are format and rate transparent.

In what follows we present an Integer Linear Programming (ILP) formulation to optimally solve the CTC problem. Since the problem is NP-complete, the ILP does not scale and its execution can take exponential time; therefore, we also provide an efficient heuristic algorithm.

4.1. ILP FORMULATION

The ILP formulation to solve the CTC is as follows.

Input and Constants:
$\boldsymbol{D}$: Traffic matrix of size *Z*x*Z* that describes the logical task-to-task communication graph in volume (bytes).
T: Number of processing elements (tasks, according to the assignment of single task per processing element) per logical cluster (LC).
K: Number of parallel OCS planes.
P: Number of ports per OCS plane per LC.
B: Constant, set to a small but positive value, in particular a value less than $\sum_n \sum_m D_{nm}$
Variables:
g_n^i: Boolean variable, equal to 1 if the task *n* belongs to the logical cluster LC_i.
g_{nm}^{sd}: Boolean variable, equal to 1 if the task *n* belongs to the logical cluster LC_s and the task *m* belongs to LC_d.
d^{sd}: Integer variable greater or equal to zero, equal to the volume (bytes) that is transferred from source LC_s to destination LC_d.
x_{ij}^k: Boolean variable, equal to 1 if an optical link is established on optical plane *k* from LC_i to LC_j, that is if optical link (k, i, j) is configured (no need to exactly specify the ports that are used to connect to LC_i and LC_j, due to the crossbar nature of the switch at each plane).
f_{ij}^{sd}: Integer variable greater or equal to zero, equal to the volume (bytes) that is transferred over an optical link from LC_i to LC_j for serving the traffic from source LC_s to destination LC_d.

$$\text{The objective is to: } \boldsymbol{Minimize} \sum_s \sum_d \sum_i \sum_j f_{ij}^{sd}$$

Subject to the following constraints:

- Cluster assignment constraints

$$\forall n : \sum_i g_n^i = 1 \qquad (1)$$

$$\forall i : \sum_n g_n^i \leq T \qquad (2)$$

$$\forall n, m, s, d : g_{nm}^{sd} \geq g_n^s + g_m^n - 1 \qquad (3)$$

- Inter-LC traffic matrix forming constraints

$$\forall s, d : d^{sd} = \sum_n \sum_m g_{nm}^{sd} \cdot D_{nm} \qquad (4)$$

- Link establishment constraints

$$\forall i, j : \sum_k x_{ij}^k \geq B \cdot \sum_s \sum_d f_{ij}^{sd} \qquad (5)$$

- Flow conservation constraints

$$\forall s, d : \sum_j f_{sj}^{sd} = d^{sd} \qquad (6)$$

$$\forall s, d, v \notin \{s, d\} : \sum_i f_{iv}^{sd} = \sum_j f_{vj}^{sd} \qquad (7)$$

$$\forall s, d : \sum_i f_{id}^{sd} = d^{sd} \qquad (8)$$

- Connectivity bound per plane constraints

$$\forall i, k : \sum_j x_{ij}^k \leq P \qquad (9)$$

$$\forall j, k : \sum_i x_{ij}^k \leq P \qquad (10)$$

Constraint (1) assigns each task to one LC. Constraint (2) limits the number of tasks in each LC to be less than *T*. Constraint (3) is used to compute the Boolean variable g_{nm}^{sd} from the Boolean variables g_n^s and g_m^d. If both g_n^s and g_m^d variables are equal to 1, then g_{nm}^{sd} takes the value of 1, otherwise it takes the value of 0 (actually, in the latter case g_{nm}^{sd} is left free, but since the minimization objective depends positively on this variable, it is directed to 0). Constraint (4) calculates the inter-LC traffic matrix ***d***. Constraint (5) establishes crossconnections from LC_i to LC_j, if at least one transmission (single-hop or multi-hop) goes from LC_i to LC_j. Constraints (6), (7), and (8) are flow conservation constraints over the OCS network. A flow that starts at the source LC_s is maintained at all intermediate hops until the destination LC_d is reached. Constraints (9) and (10) enforce the connectivity bound on the incoming and outgoing ports of an optical plane.

The solution to the CTC problem is then formed by the following optimization variables: (a) the grouping of processing elements into logical clusters (LCs) as mandated by g_i^n and (b) the configuration of the OCS network as mandated by x_{ij}^k.

4.2. *Heuristic Algorithm*

The CTC problem is computationally difficult and the above ILP formulation does not scale well with the size of the network. Thus, it is desirable to obtain efficient heuristic algorithms. In the following, we specify a heuristic for the CTC problem. We decompose the problem, by first solving the clustering problem and then the topology-configuration problem in the hybrid interconnect under study. The proposed heuristic involves three phases. In the first phase, we cluster the tasks into the logical clusters (LCs) and obtain the new traffic matrix ***d*** that captures the inter-LC traffic. The second phase takes as input the new traffic matrix ***d*** and orders the demands between LCs according to a given criterion (e.g. the communication volume). Lastly, in the third phase, a heuristic algorithm designed to sequentially serve demands is used to derive the configuration of the OCS network. The sequential algorithm processes demand pairs in the order specified by the outcome of the second phase and establishes direct OCS connections accordingly, taking into account the number *K* of parallel OCS planes and the ports *P* per optical plane. We also use Simulated Annealing (SA) to search among different demand orderings and obtain better solutions.

4.2.1. Clustering into Logical Clusters In the first phase we use spectral clustering to cluster the tasks into LCs of size *T* each. So from the traffic matrix ***D*** of size *Z*x*Z* we obtain a new traffic matrix ***d*** of size *N*x*N* ($N = \lceil Z/T \rceil$) that captures inter-LC communication traffic in bytes. The objective is to partition ***D*** such that the intra-LC communication is maximized and the inter-LC communication is minimized. Spectral graph theory provides a decent approximation, namely the spectral clustering heuristic that is widely used in clustering problems. This heuristic is based on the calculation of eigenvalues and eigenvectors and executes very efficiently even for large input matrices. We use the

2-way spectral clustering algorithm and apply it recursively [31] to hierarchically cluster the subsets until we obtain clusters of the target cardinality (T).

In particular at each iteration of the spectral clustering method we are given a traffic matrix $\boldsymbol{W}$ of N_W vertices (first iteration $\boldsymbol{W}$=$\boldsymbol{D}$, $N_W = Z$) and we calculate its normalized Laplacian matrix $\boldsymbol{L_W}$. Then we calculate the smallest nonzero eigenvalue and the corresponding eigenvector of $\boldsymbol{L_W}$ and we order that eigenvector. We then split the ordered vector in the middle and group the vertices of each half together to form two clusters. For each of these cluster we calculate its traffic matrix (take from $\boldsymbol{W}$ the rows/columns that correspond to vertices that comprise the cluster) and feed this to the next iteration of the spectral clustering heuristic. At each iteration we split one cluster into two and we do this recursively until we reach the desired cardinality of vertices *T* in a cluster.

4.2.2. Demand Ordering and Simulated Annealing The sequential demand serving heuristic algorithm of phase 3 establishes optical links for the demands of traffic matrix ***d*** by serving the demands one-by-one, in some particular order. Demands that are served earlier have higher probability of finding free resources to establish a direct optical link (single-hop transmission), as opposed to demands that are served later that may have to resort to multi-hop transmissions. Thus, the ordering in which the demands are served is important and different orderings result in different topology-configuration solutions with different performance (hop-bytes). For our study we implement the following ordering policy:

Highest Demand First (*HDF*): demands are ordered according to their communication volumes.

According to this policy, heavy demands that have high communication volumes are served first. Since the performance metric of interest is the hop-bytes, serving the demands with the highest volume over single-hop transmissions is a rational and greedy way to minimize the target metric. A number of other policies can be easily defined, based on performance targets or constraints, e.g. based on the total load from a source to all destinations, the communication degree, etc. However, since the performance depends on many parameters, it is hard to come up with an ordering policy that yields good performance across diverse inputs. Instead, we use the HDF as a baseline for creating an initial demand ordering and then optimize the ordering using the Simulated Annealing (SA) meta-heuristic. Specifically, we start with an HDF ordering and calculate its cost by sequentially serving the demands, using the heuristic described in the next subsection (this constitutes our "fitness function" in SA terminology). For a particular ordering $((s_1, d_1), ..., (s_v, d_v))$ of *v* demands, we define its neighbor as the ordering where (s_i, d_i) is interchanged with (s_j, d_j) for some *i* and *j*. Note that $v \leq N \cdot (N - 1)$, depending on the number of non-zero flows in ***d***. To generate a random neighbor, we choose the pivots *i* and *j* with uniform probability among the v demands. We use this neighbor generation procedure and the sequential demand heuristic as the fitness function in a typical SA iteration process.

4.2.3. Sequential demand serving heuristic algorithm The algorithm described here sequentially serves the traffic between the LCs described in ***d***, by configuring the OCS network to establish optical links to connect LC pairs. To keep track of the configuration of the OCS network we use two integer *K*x*N* matrices that hold the number of outgoing and incoming connections that are established for each LC at each plane. In particular, we denote by $\boldsymbol{O} = [O_{11}, O_{12}, ..., O_{1N}; ...; O_{K1}, O_{K2}, ..., O_{KN}]$, the matrix where each element O_{ki} corresponds to the number of established outgoing optical connections from LC_i at plane *k*. Similarly, we denote by $\boldsymbol{I} = [I_{ki}]$ the matrix that keeps track of the incoming connections. We also keep a set $\boldsymbol{L}$ of established optical links in the form of (k, i, j) tuples.

The demands are served sequentially in the order specified in the previous phase. For a demand, e.g. (s, d), we establish an optical link between *s* and *d* at plane *k* as long as the connectivity constraints are satisfied. We start from *k*=1 and increase it up to *K*. In particular, we check if *s* has free outgoing ports ($O_{ks} \leq P$) and also if *d* has free incoming ports ($I_{kd} \leq P$). If both are satisfied, we establish a direct optical connection between *s* and *d* by cross-connecting respective ports at plane *k* (due to the crossbar nature of the switch we can cross-connect any port from *s* to *d* on *k*th-plane). Subsequently, we increase the O_{ks} and I_{kd} elements, update the set $\boldsymbol{L}$ of established links by appending (k, s, d) to it and continue with serving the next demand. If the connectivity constraint for s and/or d is not met, we move to the next plane and if we cannot find any plane to establish the link we move to next demand. After passing all demands once, if we have more available optical ports, we repeat the same process, searching to establish two-hop connections this time, and so on until we examine four-hops (the depth that we stop is a parameter and can be controlled, but four-hops were used in our results) or the optical network is fully utilized. Once we have finished, the links to be established are given by the set $\boldsymbol{L}$. We use this as input to Johnson's algorithm [37] and compute the all-pairs shortest paths, assuming connections are served over them. The computed shortest paths are then used to calculate the average hop-bytes of the solution.

As mentioned, we additionally use Simulated Annealing (SA) to improve performance: the sequential heuristic of this phase is run multiple times for different orderings. For each ordering, a new set of established optical links and thus a new average hop-bytes value is found, and we keep the solution that yielded the fewer hop-bytes. In our implementation, we stopped SA after performing a certain number of iterations, but other stopping criteria, e.g. based on the average change of the objective, can be used.

4.2.4. Complexity of the heuristic algorithm The heuristic algorithm presented above is of polynomial complexity. The spectral clustering heuristic used in the first phase is generally considered efficient, the second phase involves the ordering of N^2 elements, while the third phase involves the execution of Johnson's algorithm that takes $O(N^2 logN + N|L|) = (N^2 logN + NKP)$ time, which is faster than Floyd-Warshall's algorithm for sparse networks like the one considered here ($K, P << N$). Lastly, if SA is used, we limit the number of its iterations, which determines the times that the third phase is executed, to stay within polynomial time bounds.

5. PERFORMANCE RESULTS

We performed experiments to estimate the performance of the proposed CTC algorithms in a hybrid EPS/OCS interconnect. We considered a number of design choices of the hybrid interconnect, in particular different levels of optical aggregation and various numbers of optical planes. We implemented both the ILP and the heuristic algorithm that solve the CTC problem as presented in Section 4. We first examine the optimality performance of the heuristic algorithm by comparing it to the optimal output of the ILP algorithm for random and small problem instances. Then we move to more realistic scenarios using only the heuristic algorithm and compare the performance of the adopted hybrid interconnect to a torus-like electronic-only interconnect. To obtain comparison results for the electronic-only interconnect we used a hierarchical mapping algorithm similar to [14]. The algorithm takes as input the logical communication graph of an application, clusters the tasks and assigns the LCs to compute resources interconnected in a fixed topology to minimize the hop-bytes of inter-LC communication. We developed both an ILP algorithm to optimally solve the corresponding mapping problem on a fixed hierarchical electronic-only interconnect and a heuristic algorithm that is also based on Simulated Annealing (SA).

5.1. Optimality evaluation of heuristic algorithm

To evaluate the efficiency of the proposed CTC heuristic presented in Section 4.2 we performed experiments to compare it to the optimal ILP algorithm of Section 4.1. Since the optimal ILP algorithm does not scale well, our comparison is done only for small logical communication graphs. Moreover, since the clustering problem is a well studied problem, we focus on the topology-configuration problem, which is the novel part of our work. To this end, we set the clustering parameter to T=1, that is, we assume that the input communication graph specifies the traffic that has to be routed over the OCS network.

To obtain the evaluation results reported in this subsection, we first created two sets of 10 logical communication graphs (traffic matrices), for N=16 and 32 end-points, respectively. The elements of the traffic matrices were chosen with 50% probability to be of zero and 50% to be of one byte

Table I. Optimality performance of the heuristic algorithms

Size	Algorithms	Hybrid interconnect K=6 Average hop-bytes	Hybrid interconnect K=6 Average running time (sec)	3D Torus Average hop-bytes	3D Torus Average running time (sec)
16	Heuristic (HDF)	149.6	0.07	237.9	0.04
	Heuristic (SA 100)	147.0	0.60	227.5	0.40
	Heuristic (SA 1000)	144.7	4.30	214.30	2.05
	ILP	144.6	6.75	222.3*	601.60*
32	Heuristic (HDF)	886.5	0.08	1253.3	0.04
	Heuristic (SA 100)	865.0	0.94	1215.8	0.61
	Heuristic (SA 1000)	839.0	8.24	1148.1	5.16
	ILP	861.1*	2399.91*	1252.3*	3600.00*

(*) set of experiments where the execution of the ILP was not completed in 1 hour for at least one instance.

volume, while the diagonal elements were set to zero, so the average connectivity was $(N-1)/2, N$ denoting the number of endpoints. The results were averaged over the 10 traffic matrices for the two different sets. We assumed that the OCS network is implemented with K=6 parallel optical planes and P=1 port per plane (equivalent to the connectivity of a fully formed 3D torus). We report the average hop-bytes performance and the average running time for the heuristic algorithm using the proposed Highest Demand First (HDF) ordering policy (actually, in these experiments with unitary demands, these are served according to their id ordering, which is the tie-breaking rule in the HDF policy) and Simulated Annealing (SA) with 100 and 1000 iterations, and the ILP algorithm. We set the maximum execution time of the ILP algorithm to 1 hour per problem instance. Experiments were run on a laptop with Intel i5 CPU at 2.5 GHz and 4 GB of memory. CPLEX 11.2 [38] was used for the ILP execution and Matlab's Simulated Annealing implementation was used for the heuristic. We also report the performance of the ILP and heuristic algorithm for hierarchical application mapping on a fixed 3D torus electronic-only interconnect. Note that because we are examining small problem instances, a 3D torus cannot be fully formed with the number of end-points considered (e.g. with 16 end-points we end up with a torus of connectivity degree equal to 4). So in this set of results we cannot fairly compare the hybrid to the 3D torus interconnect. However, we report on the results of the 3D interconnect to evaluate the optimality performance of the related heuristic that we are going to use to estimate the performance of such interconnects in the following sections. We summarize our results using the above setting in Table I.

Using the optimal CTC ILP algorithm we were able to track optimal solutions for 16 end-points within a few seconds. However, for 32 end-points, we were not able to obtain optimal solutions for all instances within the time limit of 1 hour per instance that we set. From Table I, we observe that the the CTC heuristic performed remarkably well in this small scale experiments. For 16 end-points the heuristic found solutions very close to the ILP and its performance depends on the number of different orderings (iterations) that are examined by the SA. The case where we use the heuristic with only a single ordering (the HDF ordering) without employing SA, has obviously the worst hop-bytes performance. The performance of the heuristic improves as we increase the number of examined orderings, approaching optimal performance for 1000 SA iterations. However, as expected, the running time of the heuristic increases as the number of SA iterations increases. The running time of SA with 1000 iterations for 16 end-points is comparable to that of the ILP algorithm. However, the bad scalability performance of the ILP algorithm starts to become apparent in the 32 end-point experiments and its average running time increases rapidly. In the majority of the examined instances of this size, the ILP algorithm did not succeed in finding optimal solutions within one hour, in which case we report the best solution found up to the time it was stopped. Since a number of experiments finished in less than 1 hour, the average running time reported is lower, but still for this set of experiments we were not sure if we had found the optimal solution in all experiments. It turned out that the obtained solutions were sub-optimal, since the heuristic algorithm with 1000 SA iterations was able to find better solutions. In contrast to the ILP, the heuristic scales well, and even for 1000 SA iterations on 32 end-points, its average running time is remarkably low. Again, the performance of the heuristic improves as we increase the number of SA iterations, while the average running time increases as well. Thus, using SA we can control the tradeoff between running time and performance. At least for these small size experiments, the results show that even with a few SA iterations (e.g. 100) we can have near optimal performance in very low execution time. With respect to the mapping problem on a fixed 3D torus electronic-only interconnect, from Table I we observe that the developed heuristic exhibits very good performance. The mapping problem on fixed interconnects seems to be harder and the related optimal ILP algorithm requires higher running times to find the solutions, when compared to the optimal CTC ILP algorithm. In particular, for the 16 end-points the ILP algorithm failed to find the optimal solution for some of the examined instances so the reported solutions are not optimal. This was not the case for the hybrid interconnect. For 32 end-points, all examined problem instances could not be solved within the 1 hour limit, while the heuristic algorithm was able to outperform it, even with just 100 SA iterations.

5.2. *Performance evaluation with real input*

We now advanced to larger-scale and more realistic experiments. We use the proposed CTC heuristic to calculate the hop-bytes and through this metric estimated the performance of the hybrid interconnect and compared it to the related performance of electronic-only interconnects. To this end, we first captured the logical communication graphs of two representative HPC kernels. In particular, we evaluated SuperLU [32] and FFTW [33] on an HPC cluster located at IBM Dublin. SuperLU performs LU factorization of a sparse matrix, and FFTW performs forward and inverse Fast Fourier Transformations. Any MPI application, such as the ones examined, comprises processes (tasks) running in parallel each one corresponding to what is called an MPI-rank. Note that an application can be instantiated to use different numbers of MPI-ranks. Thus, the number of MPI-ranks is the number of parallel tasks which also gives the number of cores that are used (assuming the pinning of one task to one core which is the common practise and the one used in our experiments). We used the IPM monitoring tool [33] to capture the MPI task-to-task traffic that is generated by these workloads. The applications were run with 240, 480, 960, and 1920 MPI-ranks. Fig. 3 presents the task-to-task communication graphs obtained after a single run of each of the tested applications with 480 MPI-ranks. Note that the applications used are well studied applications, known to exhibit static communication graphs [24],[25], and fall within the static application category, as described in Section 1. We also executed them a number of times and visualized their communication graphs to verify that they exhibit similar patterns irrespective of the input. The captured task-to-task communication graphs were used as input for evaluation of our approach, as well as for performing hierarchical mapping on the electronic-only interconnect. Given the lack of a widely-accepted method to scale traffic, we extrapolated to higher scales by producing synthetic traffic matrices that were isomorphic to the captured ones to examine problem executions that we were not able to perform on our cluster.

We assumed that the hierarchy of the system consists of 12 cores per node and 40 nodes per rack (total of 480 cores per rack), driven by the specifications of the cluster at IBM Dublin. We report results for the case where the OCS network is either connected directly to the compute nodes or to top-of-rack switches, and for K=4, 6 and 8 parallel OCS planes and P=1 port per plane. For comparison purposes, we also estimated the performance of electronic-only 2D-, 3D-, and 4D-torus interconnects, (thus having the same connectivity degrees as the corresponding OCS networks). For both hybrid and electronic-only interconnects we used 100 SA iterations.

Table II reports results for SuperLU, assuming that the OCS network is connected directly to the compute nodes; as a comparison, we also report equivalent results for a torus EPS-only network formed by the compute nodes. More specifically, Table II reports the results using the communication graphs that were captured by executing SuperLU to factorize the “webbase-1M” matrix taken from [34], on 240,480, 960, 1920 MPI-ranks, which corresponds to 20,40, 80 and 160 nodes, respectively (assuming pinning of a rank to a processor core). Note that we used Mbytes to measure the volume of data and thus the values reported in the table are measured in hop-Mbytes.

From Table II we observe that the hybrid interconnect exhibits lower hop-Mbytes than the electronic-only interconnect. Even a hybrid interconnect with K=4 parallel optical planes has lower hop-Mbytes for almost all problem instances in comparison to an electronic-only interconnect with

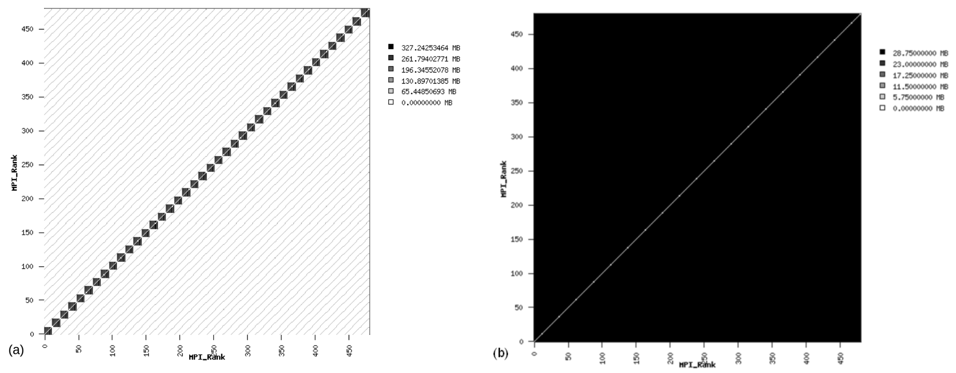

Figure 3. Captured logical communication graphs of (a) SuperLU and (b) FFTW on 460 MPI-ranks.

Table II. Performance (hop-Mbytes) for captured communication graph of SuperLU using the ”webbase-1M” matrix as input

Input	Hybrid interconnect hop-Mbytes			Electronic-only interconnect hop-Mbytes		
	K=4	K=6	K=8	2D	3D	4D
20 nodes(1/2 rack)	5235	4046	3730	6973	5604	5604
40 nodes (1 rack)	12703	9944	8683	13537	13272	12724
80 nodes (2 racks)	16164	12622	11136	23514	16784	16481
160 nodes (4 racks)	37868	27381	23861	60685	41362	36965

double connectivity (4D torus). Note that the improvement that we obtain when using the hybrid interconnect does not come from the highest capacity supported by the OCS network. Instead, the improvement comes from the configurability of the OCS part of the hybrid interconnect that was exploited to create more efficient topologies to serve the communication of the application.

Fig. 4 presents the relative hop-bytes improvement achieved by our approach applied in a hybrid interconnect over a 3D-torus electronic-only interconnect. Both captured (“webbase-1M” matrix) and synthetically generated traffic results are depicted. By comparing results for 20 up to 160 nodes we observe that the improvement we got using the synthetically generated traffic is in-line with the ones obtained with real input. As expected, the improvement is higher as we increase the number of parallel OCS planes. A 3D torus network cannot be fully constructed on a low number of nodes (up to a few tens of nodes, e.g. 40), so the improvement obtained by the hybrid interconnect as opposed to the 3D torus networks for problems of that size is partially explained by the 3D torus deficiency. Otherwise, in small size networks, where the network is dense (the connectivity degree is comparable to the number of end-nodes), the performance gains that can be obtained by optimizing the topology are not significant. Thus, for small size networks, a fair comparison would give lower improvement than the one observed in Fig. 4. However, as the number of nodes increases, the topology starts to play a more important role and thus we expect to obtain higher improvement. In particular, as the number of nodes increases up to 80 nodes, we have two opposing factors: on the one hand, a fully constructed torus topology that improves the performance of the 3D torus network as opposed to lower size 3D torus instances, and on the other hand the proposed CTC heuristic that creates topologies whose efficiency improves as the number of nodes increases. These two factors balance each other, resulting in almost constant hop-bytes improvements for networks up to 80 nodes. At 80 nodes and above, the 3D torus is fully constructed and the comparison becomes fair. For these problem sizes, we can verify that the improvement obtained by using the hybrid interconnect as opposed to a 3D torus network increases as we move to networks with more nodes. For K=6 (hybrid interconnect has equal connectivity degree to the 3D torus), we observe that the improvement is approximately 24% when executing SuperLU on up to 80 nodes, and increases up to 31% and 42% when the application is executed on 160 and 320 nodes, respectively.

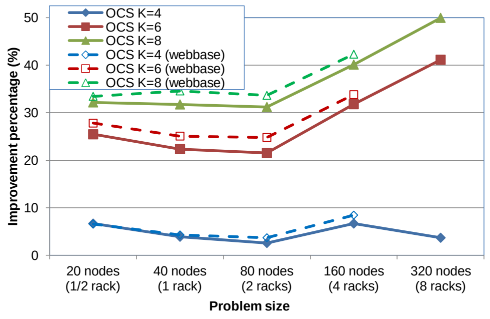

Figure 4. Percentage of hop-bytes improvement of SuperLU, when using the hybrid interconnect over a 3D torus electronic-only interconnect, using compute nodes as the aggregation level.

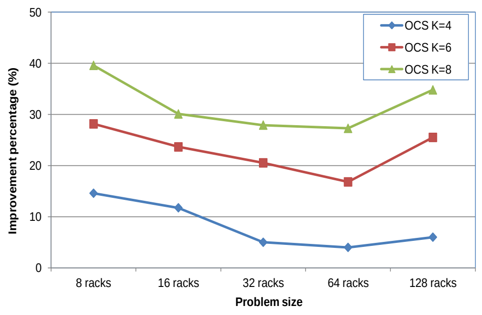

Figure 5. Percentage of hop-bytes improvement of SuperLU, when using the hybrid interconnect over a 3D torus electronic-only interconnect, using racks as the aggregation level.

Fig. 5 presents the improvement over a 3D electronic-only interconnect topology for the case where the OCS network is connected to top-of-rack switches. Note that in this set of experiments we neglected the performance of the network that interconnects nodes within a rack, and we only consider inter-LC (or in this particular case inter-rack) traffic. We report results using synthetically generated traffic to emulate the execution of SuperLU from 8 up to 128 racks. As discussed above, for small scale problem instances, a 3D torus cannot be fully constructed. However, for larger scale-out instances that we have a fair comparison, we see that the hybrid interconnect outperforms the 3D torus network and we have up to 25% better hop-bytes performance than the electronic-only interconnect for 128 racks and equal connectivity degree (K=6). Similar improvements were observed for FFTW application that are not reported here due to space limitations.

5.2.1. Cost considerations In addition to performance evaluation, we also created simple models to estimate the capital expenditure (CAPEX) cost of the hybrid interconnect and compare it to alternative electronic-only solutions. Due to extreme price volatility in an evolving technology domain, any specific price trend assumption would be speculative and may not survive over time. Instead, we conducted a cost comparison based on the number of ports and transceivers (TxRx) required in each case. Assuming rack aggregation level, the number of EPS ports utilized in the hybrid interconnect is $N^H_{Eports} = P \cdot K \cdot N + M \cdot N + R \cdot N$ where R is the number of electronic ports of the edge that are connected to the compute nodes (assuming R compute nodes per rack), and the number of OCS ports is $N^H_{Oports} = P \cdot K \cdot N$. Assuming that optical TxRx are used for the EPS core and OCS network, the number of TxRx required for the hybrid interconnect are: $N^H_{TxRx} = P \cdot K \cdot N + M \cdot N$. Note that the price of a fiber cable is at least one order of magnitude lower than that of an optical transceiver, an electronic or an optical port. So it stands to reason that the total cost of the fibers in the system is low and thus is not considered in our calculations. For an XD torus electronic-only network (X=2,3,...), the number of EPS ports is $N^E_{Eports} = 2 \cdot X \cdot D + R \cdot N$ and the number of TxRx is $N^E_{TxRx} = 2 \cdot X \cdot N$, assuming that only inter-rack communication is performed with optical TxRx. We found the relative cost of an optical to electronic port to be currently in the [0.4-0.6] interval, assuming 10Gbps Ethernet switching technology.

Assuming that the EPS-core network of the hybrid interconnect is very basic and inexpensive, $M << 2 \cdot X$, and for a ratio of optical to electronic port equal to 0.5, the hybrid interconnect with K=4 and P=1, has equivalent cost to a 3D torus network. As we saw earlier, a hybrid interconnect with these specifications was able to outperform in terms of hop-bytes the corresponding 3D electronic-only network. The prices of OCS switches will tend to fall rapidly, as high as 80% [35], as vendors begin widely deploying this technology in DCs/HPC clusters, so higher degree hybrid interconnect could be deployed cheaper, obtaining more significant improvements.

A benefit of the proposed hybrid interconnect that is not factored into the cost model is that it is future-proof, since the OCS network is agnostic to protocols, modulation formats and data rates. Thus, to increase the capacity of the hybrid interconnect system we only need to upgrade the electronic-edge that accesses the OCS network. This also indicates that the cost of the OCS port and that of the hybrid network will almost certainly not increase, while the cost of the electronic-only interconnect might increase rapidly, especially if we move to a newer technology with higher bandwidth (e.g. 40 or 100Gbps). Lastly, note that the optical technology, due to its transparent nature, consumes much less energy compared to active electronic switching, a cost-saving factor in the operational expenditure (OPEX) not captured in our above model.

6. CONCLUSIONS

We presented methods to cluster the tasks and identify the topology configuration of the (reconfigurable part) of a hybrid EPS/OCS interconnection network to efficiently serve the task-to-task traffic produced by a parallel application with known logical communication graph. In particular, we provided an optimal ILP formulation and a heuristic algorithm that is based on Simulated Annealing. The methods presented are general and can be used in many different settings, irrespective of the level at which the optical network is connected and the number and ports of optical switches employed. After evaluating the optimality of the proposed heuristic, we used it to estimate the performance of the target hybrid interconnect and compare it against application mapping on conventional fixed, electronic-only interconnects that are based on toroidal topologies. Our results indicated that the hybrid interconnect can exhibit better average hop-bytes performance than torus interconnect with higher connectivity degrees. By evaluating networks with the same connectivity degree we observed hop-bytes improvements as high as 42%. Subject to the relative costs of the electronic and optical ports, these can come at a comparable cost, while the data rate agnostic and transparent nature of optical technology ensures better upgradability and lower power consumption for the hybrid interconnect.

ACKNOWLEDGEMENT

This work has been partially supported by Industrial Development Agency (IDA) Ireland and the Irish Research Council for Science, Engineering and Technology (IRCSET).

REFERENCES

1. A. Geist, "Paving the roadmap to exascale", SciDAC Review, Special Issue on Information Technology in the Next Decade, vol. 16, pp. 52-59, 2010.
2. R. Brightwell, B.W. Barrett, K. S. Hemmert, K. D. Underwood,"Challenges for High-Performance Networking for Exascale Computing", Conference on Computer Communications and Networks (ILCCN), 2010.
3. Cray Inc. Cray XE6 http://www.cray.com/Assets/PDF/products/xe/CrayXE6Brochure.pdf.
4. Y. Ajima, S. Sumimoto, T. Shimizu, "Tofu: A 6d mesh/torus interconnect for exascale computers", Computer 42(11), 2009.
5. D. Chen, N.A. Eisley, P. Heidelberger, R.M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D.L. Satterfield, B. Steinmacher-Burow, J.J. Parker, "The IBM Blue Gene/Q interconnection network and message unit", Supercomputing, 2011.
6. S. H. Bokhari, "On the Mapping Problem", IEEE Transactions on Computers, 30(3), 1981.
7. S. W. Bollinger, S. F. Midkiff, "Heuristic Technique for Processor and Link Assignment in Multicomputers", IEEE Transactions on Computers, 40(3), 1991.
8. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. Giampapa, M. C. Pitman, "Blue matter: Approaching the limits of concurrency for classical molecular dynamics", Supercomputing, 2006.
9. P. Balaji, R. Gupta, A. Vishnu, P. Beckman, "Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems", Computer Science Research and Development, 26(3-4),2011.
10. G. Bhagnot, A Gara, P. Heidelberger, E. Lawless, J. C. Sexton, R. Walkup, "Optimizing task layout on the Blue Gene/L supercomputer", IBM Journal of Research and Development, 2005.
11. H. Yu, I.H Chung, J. Moreira, "Topology mapping for Blue Gene/L supercomputer", SuperComputing, 2006
12. A. Bhatele, G.R. Gupta, L. V. Kale, I.H. Chung, "Automated Mapping of Regular Communication Graphs on Mesh interconnects", Conference on High Performance Computing (HiPC), 2010.
13. A. Bhatele, L. V. Kale, "Heuristic-Based Techniques for Mapping Irregular Communication Graphs to Mesh Topologies", Conference on High Performance Computing and Communication (HPCC), 2011.
14. I-H Chung, C-R Lee, J Zhou, Y. C. Chung, "Hierarchical Mapping for HPC Applications", International Parallel and Distributed Processing Workchop (IPDPSW), 2011.

15. Al-Fares, Loukissas, Vahdat, A Scalable, "Commodity Data Center Network Architecture", SIGCOMM, 2008.
16. T. Benson, A. Akella, D. A. Maltz, "Network traffic characteristics of data centers in the wild", Internet Measurement Conference (IMC), 2010.
17. A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, S. Sengupta, "VL2: A Scalable and Flexible Data Center Network", SIGCOMM, 2009
18. K. J. Barker, et. al., "On the Feasibility of Optical Circuit Switching for High Performance Computing Systems", Supercomputing, 2005.
19. K. J. Barker, D. J. Kerbyson, "Performance Analysis of an Optical Circuit Switched Network for Peta-Scale Systems", European Conference on Parallel and Distributed Computing (EuroPar), 2007
20. L. Schares, et. al., "A Reconfigurable Interconnect Fabric with Optical Circuit Switch and Software Optimizer for Stream Computing Systems", Optical Fiber Communications Conference (OFC), 2009.
21. N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman,G. Papen, A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers", SIGCOMM, 2010.
22. G. Wang, D. Andersen, M. Kaminsky,K. Papagiannaki, T. S. E. Ng, M. Kozuch, M. Ryan, "c-Through: Part-time Optics in Data Centers", SIGCOMM, 2010.
23. K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, Y. Chen, "OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility", Symposium on Networked Systems Design and Implementation (NSDI), 2012.
24. K. Asanovic, et. al., "The Landscape of Parallel Computing Research: A View from Berkeley", tech. report, 2006.
25. S. Kamil, L. Oliker, A. Pinar, J. Shalf, "Communication Requirements and Interconnect Optimization for High-End Scientific Applications", IEEE Transactions on Parallel and Distributed Systems, 21(2), 2009.
26. D. Ajwani, S. Ali, K. Katrinis, C-H Li, A. Park, J. Morrison, E. Schenfeld, "A Flexible Workload Generator for Simulating Stream Computing Systems", MASCOTS, 2011.
27. Integrated Performance Monitoring (IPM): http://ipm-hpc.sourceforge.net/
28. K. Christodoulopoulos, K. Katrinis, M. Ruffini, D. O'Mahony, "Topology Configuration in Hybrid EPS/OCS Interconnects", European Conference on Parallel and Distributed Computing (EuroPar), 2012.
29. X.J. Zhang, R. Wagle, J. Giles,"Vlan-based routing infrastructure for an all-optical circuit switched lan", IEEE conference on Global telecommunications, (GLOBECOM), 2009
30. V. Liberatore, "Circular Arrangements", Colloquium on Automata, Languages and Programming (ICALP), 2002.
31. J. Shi, J. Malik, "Normalized Cuts and Image Segmentation", IEEE Transactions on PAMI, 22(8), 2000.
32. Super-LU: http://crd-legacy.lbl.gov/ xiaoye/SuperLU/
33. FFTW: http://www.fftw.org/
34. http://www.cise.ufl.edu/research/sparse/matrices/
35. http://www.lightreading.com/document.asp?doc_id=213809&f_src=lightreading_gnesws
36. Polatis DirectLight, http://www.polatis.com
37. D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks", Journal of the ACM, 24(1), 1977.
38. http://www-01.ibm.com/software/integration/optimization/ cplex-optimizer/

Appendix: Complexity of the topology-configuration problem

The topology-configuration problem with multi-hop transmissions (abbreviated TC) at a single optical plane is defined as follows. We are given a fully connected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, with $\boldsymbol{V}$ being the set of nodes and $\boldsymbol{E}$ the set of candidate directed links that connect every pair in $\boldsymbol{V}$. We are given a traffic matrix $\boldsymbol{d}$ that describes the weight of communication between the nodes in $\boldsymbol{V}$. We want to identify the set $\boldsymbol{L} \subseteq \boldsymbol{E}$ of uni-directional links to establish, with the constraint that the created graph $\boldsymbol{G}' = (\boldsymbol{V}, \boldsymbol{L})$ has connectivity degree less or equal to P. The objective is to transfer $\boldsymbol{D}$ over $\boldsymbol{G}'$ and minimize the weighted sum of the traversed hops. The weight is given by the traffic in $\boldsymbol{D}$ and the hops are calculated considering that communication is performed over shortest paths in $\boldsymbol{G}'$.

To prove the NP-completeness of the TC problem, we reduce the circular arrangement (CA) problem that is known to be NP-complete [31] to the TC. The CA problem is defined as follows. We are given a set of nodes $\boldsymbol{V}$ and a traffic matrix $\boldsymbol{W}$ and we want to arrange the nodes on a directed circle, that, is to embed the graph $\boldsymbol{W}$ on a directed circle, and to minimize the total weighted arc hop length. We focus on the directed CA problem, but both directed and undirected versions are considered in [31].

For an instance of the Circular Arrangement (CA) problem, we can construct a corresponding instance of the TC problem in polynomial time. We set the traffic matrix $\boldsymbol{D} = \boldsymbol{W} + \boldsymbol{A}$, where $\boldsymbol{A}$ is a matrix with all ones except for the diagonal elements that are set to 0, and $\boldsymbol{W}$ is the weighted graph that is the input to the CA. We set the connectivity degree bound of the TC to P=1. Remember that P corresponds to the number of undirected links of each node. We claim that the optimal solution ***G'*** of this TC problem is also the optimal solution of the corresponding CA problem. The graph ***G'*** that is created by solving the TC problem is a circle connecting all nodes. This is because the matrix $\boldsymbol{A}$ that was added on $\boldsymbol{W}$ to obtain $\boldsymbol{D}$ introduces traffic between all pair of nodes so the final topology has to be a connected graph. A circle is the only topology with connectivity degree 1 that is also a connected graph (to be more specific, if it is a line, we can connect the two edge nodes and obtain a circle with the same cost). Since the topology ***G'*** is a circle, the shortest path over it corresponds to the hops of the circle arcs. Note also that the matrix $\boldsymbol{A}$ introduces a constant to the objective cost function, since serving $\boldsymbol{A}$ has equal cost in every circular arrangement. Thus, matrix $\boldsymbol{A}$ is added to $\boldsymbol{W}$ to make the final topology of the TC problem a circle and contributes a constant on the cost objective, which can be substracted at the end. Thus, the optimal solution of the TC problem gives us an optimal solution of the CA problem. QED