On the Design of Clean-Slate Network Control and Management Plane

Hammad Iqbal [†], Taieb Znati [‡]
{hiqbal, znati}@cs.pitt.edu

[†‡] School of Information Sciences
[‡]Department of Computer Science
University of Pittsburgh, USA
Pittsburgh, PA 15260

PITT CS TR-09-168

August 2009

Abstract

We provide a design of clean-slate control and management plane for data networks using the abstraction of 4D architecture, utilizing and extending 4D's concept of a logically centralized Decision plane that is responsible for managing network-wide resources. In this paper, a scalable protocol and a dynamically adaptable algorithm for assigning Data plane devices to a physically distributed Decision plane are investigated, that enable a network to operate with minimal configuration and human intervention while providing optimal convergence and robustness against failures. Our work is especially relevant in the context of ISPs and large geographically dispersed enterprise networks. We also provide an extensive evaluation of our algorithm using real-world and artificially generated ISP topologies along with an experimental evaluation using ns-2 simulator.

This research was sponsored by the NSF under ITR Award ANI-0426886.

On the Design of Clean-Slate Network Control and Management Plane

I. Introduction

Present day data networks are controlled by a variety of distributed routing algorithms (e.g. OSPF, IS-IS, BGP, etc.), each working independently to achieve some network-wide objective, while operating collectively on diverse physical network devices. This has created a situation where each network function (e.g. inter-domain and intra-domain routing) maintains a distinct state across many different physical devices and is governed by its own set of configuration rules and protocol logic, making it extremely difficult to control their interactions. Consequently, the management of typical data networks requires extensive manual configuration of individual protocol parameters, leaving the networks fragile [1]–[3] and insecure [4].

Effective control and management is especially a challenge for large and geographically dispersed networks, such as first and second tier ISPs, where it is important to efficiently manage the network resources across a large number of heterogeneous network devices, while meeting strict constraints on network availability and reliability. The control of such networks has additional challenges as the robustness, scalability and responsiveness of the control functionality is impacted by scale and geographical dispersion.

Incremental solutions to improve network management, including the use of better management tools, have been ineffective as they try to match the pace of changes in various device operations and technical advances. Additionally, newer services and objectives beyond best-effort routing place new demands on the network control algorithms that are difficult to meet in the presence of intricate inter-dependencies and distributed state and logic. This often necessitates the error prone and laborious process of indirectly inducing desired behavior in dynamic protocol operation through static configurations, e.g. traffic engineering [5].

To tackle the challenge of management complexity, an alternative approach to incremental solutions involves centralization of control state and logic. This approach is the basic tenant of the 4D architecture [2]. The 4D architecture advocates a new layering design of the IP networks which separates the task of packet forwarding, a data layer function, from the task of network control, an operation and management function. This separation of data and control layers is in contrast with the current practice where the data forwarding mechanism and control logic are intertwined inside monolithic network devices, such as network routers or switches. This approach to network control necessitates the centralization of control state and logic inside a *logically centralized* Decision plane, that is responsible for collecting, computing, and maintaining the state required by the network devices to operate.

The design of an efficient and robust Decision plane requires careful consideration of the design efficiency and robustness. A physically centralized decision plane design was investigated in [6], [7] where replication of physical Decision Elements (DE) was used to ensure Decision plane robustness to DE failures. An alternative design approach was identified in [8], where the logical Decision plane was distributed over physically independent DEs. In this design, each DE controls a subset of the whole network, and works collaboratively with other DEs to achieve overall network control. However, it is also important to ensure that the reliability of the physically distributed control approach matches or exceeds the reliability offered by today's distributed architecture.

In this paper, we focus on the design of logically centralized clean-slate Decision plane using the 4D clean-slate network paradigm as the basis for developing an efficient, robust, and reliable network control architecture. We argue that the Decision plane design should be based on meeting the following objectives:

- **Scalability:** The Decision plane must be scalable to network size in terms of the number of routers;
- **Robustness:** The design should be dynamically adaptable to failures at both Decision and Data planes.
- **Optimal convergence:** Total response time of the Decision plane to any event must be minimized, and the protocol operating at the Decision plane should be able to converge quickly enough to operate on the time-scale of events happening at the Data plane, e.g. router/link failures.

Achieving these objectives requires the development of a Decision Plane Protocol (DPP) that maintains a network-wide state across the set of physically distributed DEs, and presents a uniform interface to the network switches or routers[1]. Furthermore, the DEs and their assigned routers must respond swiftly to events such as failures and traffic surges. This requires that the delay between the DEs and their assigned routers be minimized. This paper addresses these design requirements and presents a Decision plane where a set of DEs, each governing a subset of routers, collaboratively maintain a network-wide state to support network-wide routing decisions.

Our work is especially relevant in the context of ISPs and similar large and geographically dispersed networks, where

[1] We use "router" as a generic label for routers or switches in the 4D Data plane, while "DE" is used to represent Decision Elements in the 4D Decision plane

network-wide control is highly desired but any Decision plane design must meet stringent challenges of scalability and robustness, which are explicit design objectives of our scheme. In our design, an individual member of the Decision plane only governs a subset of the total number of routers in the Data plane, and Points Of Presence (POP) in ISP topologies are naturally amenable to such grouping.

The main contribution of our work is the design of a scalable *logically centralized and physically distributed* Decision plane. The first building block in our design is the formulation of an optimization problem focused on efficient assignment of routers to DEs. The solution of this problem leads to an algorithm that minimizes network delay between the DEs and their assigned routers while balancing the load at the DEs. This algorithm is then used in the proposed protocol that is responsible for the operation of logically centralized Decision plane. Our paper is organized as follows: We describe the network model used in our paper in §II. Tradeoffs in the design of assignment algorithm are considered in §III. In §IV, formulation of the router assignment problem is presented along with a novel adaptive algorithm for its solution. §V describes our proposed protocol for logically centralized Decision plane operation. §VI provides an analysis of the algorithm's performance on real world and artificially generated topologies and §VII describes the simulation results of our Decision plane implementation. We explore related work in §VIII and §IX concludes our paper.

II. Network Model

We utilize the abstraction for 4D architecture [2], which decomposes a data network into four separate planes viz. Data, Discovery, Dissemination, and Decision plane. This layering provides a separation of the data forwarding mechanisms, such as packet forwarding and filtering, from the state and logic required to manage the network. The Decision plane of this architecture is therefore responsible for maintaining information about the state of network devices and utilizing this centralized view for computing the mechanisms (such as routing tables) that are required by the network devices. However physical centralization of the network control logic is undesirable to avoid potential problems with scalability and fault-tolerance. Logical centralization of network control is an alternative, explored earlier in [8] that proposed using a set of Decision Elements (DEs) which can collaborate to perform the function of network-wide control, adding a level of distribution in the Decision plane. We model the Decision plane in this paper utilizing the same abstraction of logical centralization; the high-level design of which is exemplified in Fig. 1 for an ISP topology spanning the continental US with several POPs. The figure also illustrates the few basic assumptions taken in our network model.

1) The entire network topology is under a single administrative control.
2) The Decision plane is fully connected, i.e. there is a path from each DE to all other DEs that is not dependent on the operation of Data plane.
3) Positioning of DEs corresponds to the natural geographical clustering of routers in the Data plane, e.g. within an ISP POP.

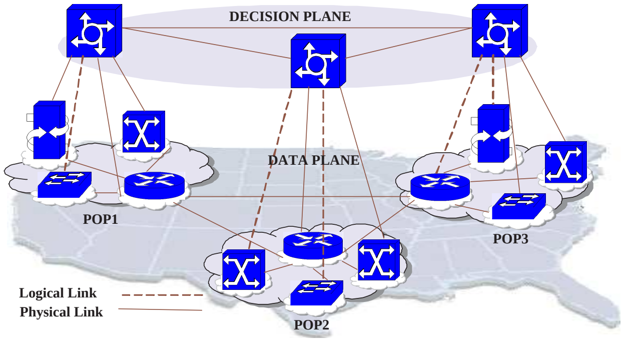


Fig. 1. Overview of the Decision plane design

We believe these assumptions are easy to meet in any reasonably large network where control and management is presently an issue. The first assumption is necessary for consistent network-wide management and deserves no further explanation. The use of dedicated out-of-band control paths in the second assumption is in contrast with the in-band paths used in current IP networks, where data and routing information packets share the same channels. Although it is possible to use the same scheme in logically centralized Decision plane design, we have purposely avoided the potential complexity and network fragility introduced by piggybacking control information over data paths. Our use of out-of-band paths is analogous to the SS7 signaling used in PSTN networks [9] and can be similarly implemented. Use of separate time-slots or wavelength channels for control messages is one way this separation could be accomplished. Finally, our third assumption positions DEs in accordance with the clustering of routers in the underlying Data plane, using the techniques discussed in [8]. This ensures that latency of Decision plane response, and convergence delay in case of failures, is kept close to minimum.

In our design, each DE is only responsible for computing routing tables for the routers under its direct control, i.e. a subset of the total number of routers in a network. We refer to this (sub)set of routers as an *area* and it marks the extent of a DE's direct control over the network. Moreover, DEs exchange reachability information about their areas and utilize this information in establishing routing paths between different areas. In the case of shortest-paths routing, which we employ for route computation, a path between routers in two different areas must travel the inter-area links between them. This results in optimal routes only under the condition that a similar routing process on the complete topology would have selected the same path. Similar argument also applies to the intra-area routes. It is easy to see that this condition is fulfilled in topologies where distances between routers inside geographical clusters are less than the distance between the clusters. We believe network size and geographical distances between sub-entities in enterprise and ISP networks naturally allow the fulfillment of this condition.

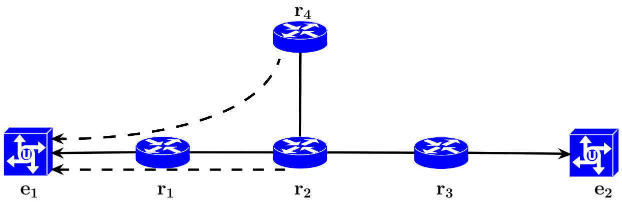


Fig. 2. Effect of contiguity constraint on a sample topology where multi-hop router assignments are indicated by dashed lines. The (infeasible) assignment of router r_4 to DE e_2 would have resulted in minimal delay and optimal load-balancing.

III. Trade-offs in Decision Plane Design

Robustness of the Decision plane is dependent on the mechanisms employed to ensure its continued functioning in case of failures. While the Decision plane routing logic deals with failures happening at the Data plane, the mitigation of failures at the Decision plane is dependent on its own design. An approach to this problem was presented in [7], where the Decision plane was designed to be physically centralized and multiple hot-standby DEs were used to increase its robustness in case the current "master" DE fails.

In contrast, a DE in a logically centralized Decision plane is not required to control the entire AS; only a subset of the total number of routers are under the control of a single DE. Any DE failure would therefore orphan the routers under its control. This calls for a scheme that reassigns orphaned routers to the functional DEs so that network control is reinstated.

This assignment of routers takes place both at network bootstrap and as a result of DE failures. It involves a trade-off in minimizing routing convergence delay, response time, and load balancing at the Decision layer. The routing convergence delay — transient time period between DE failure and orphaned routers' reception of new routing assignments — represents loss of management control over the orphaned devices, and must be minimized. Similarly, in normal operation the response time of Decision plane also needs to be minimized. In both cases, aggregate router-DE delay provides a natural metric for the minimization objective. Additionally, large variation in DE loads must be avoided as it can result in slower Decision plane response in parts of the networks and increased potential for DE failures.

Assignment mechanism is also constrained in a unique way as any router assignment must adhere to the underlying physical data plane topology. Specifically, since a DE only controls the routers in its own area, the assignment mechanism must avoid any assignment that involves the usage of inter-area paths between routers belonging to the same area. This condition is necessary to ensure that routers in an area can be governed locally without requiring global network knowledge. Therefore, there must be a physical path between routers that are assigned to the same DE that does not involve any links or routers not totally contained within the same area. We refer to this condition as the contiguity constraint and Fig. 2 illustrates a simple example where the assignment that is optimal in terms of delay and load balancing objectives does not satisfy the contiguity requirement.

Trade-offs also exist between complexity of a recovery scheme and the desired level of robustness. For example, we can generalize a simple scheme of using backups as proposed in [7], [10], where each router is statically configured with a primary and an ordered list of standby DEs. Failure of the primary DE automatically results in the assignment of its orphaned routers to their highest-ranked functional DEs. However, it is easy to show that this scheme can lead to uneven DE workloads in case of multiple DE failures, potentially causing severe performance degradation. Therefore, we note that while fixed ordering schemes may work for single DE failure scenarios, it is desirable to have an adaptive mechanism, that can assign orphaned routers to feasible DEs while, 1., balancing the DE workload and, 2., minimizing the propagation delays between routers and DEs. In the following section we describe our design of such adaptive router assignment mechanism.

IV. Adaptive Assignment of Data Plane Devices

Let $R = \{r_1, r_2, ..., r_m\}$ be the collection of routers in a AS, assumed to be homogeneous in terms of their demands of Decision plane resources, and $E = \{e_1, e_2, ..., e_n\}$ be the set of n functional DEs in the network. For any r_i, $N(r_i)$ denotes the set of routers in physical open neighborhood of r_i, i.e. r_i and all of its physically adjacent routers. We define $A(e_j)$ to be the set of routers assigned to e_j and A as the adjacency matrix of router assignments for all DEs in E, which is the output of the assignment problem. Let $x(r_i, e_j)$ be a binary indicator variable defined as $x(r_i, e_j) = 1 \iff e_j \leftarrow r_i$. Let $d(r_i, e_j)$ be the minimum delay between router r_i and a DE e_j, and $D[d(r_i, e_j)]_{m\text{xn}}$ be the matrix of all such delays. Let $L_j = \sum_{r_i \in R} x(r_i, e_j)$ be the load on DE e_j and Q_j be the capacity, i.e. the maximum number of routers, that e_j is able to govern.

We assume that information about the network topology, specifically router adjacencies and delay, would be available to the Decision plane as part of the service offered by the Discovery and Dissemination planes of 4D architecture. Use of source routes [7], [10] is one method by which such information can be collected, and §V-B discusses the protocol primitives that can be used for inter-layer communication. However, the design specifics of Discovery and Dissemination planes are beyond the scope of this work.

A. ILP Formulation

From the discussion of the previous section, the objective of the assignment problem is to assign routers in R to DEs in E in such a way that aggregate delay between routers and their assigned DEs is minimized, while ensuring that the DE workload is balanced. Formally, we define our objective function as $\sum_{e_j \in E} \sum_{r_i \in R} d(r_i, e_j) x(r_i, e_j)$ and introduce a constraint to balance the loads using the average load L_{avg}, and a load balancing parameter $\Delta \geq 1$.

$$L_{avg} = m / \sum_{e_j} Q_j \qquad 0 < L_{avg} \leq 1$$

The optimization problem can be formulated as the following ILP:

$$\text{Minimize} \sum_{e_j \in E} \sum_{r_i \in R} d(r_i, e_j) x(r_i, e_j) \qquad (1)$$

s.t.

$$\sum_{e_j \in E} x(r_i, e_j) = 1 \qquad {\scriptstyle \forall r_i \in R} \qquad (2)$$

$$\sum_{r_i \in R} x(r_i, e_j) - Q_j \leq 0 \qquad {\scriptstyle \forall e_j \in E} \qquad (3)$$

$$L_j \leq \lceil \Delta L_{avg} Q_j \rceil \qquad {\scriptstyle \forall e_j \in E} \qquad (4)$$

$$\sum_{r_k \in N(r_i)} x(r_k, e_j) \geq x(r_i, e_j) \qquad {\scriptstyle |A(e_j)| \geq 1, \forall r_i \in R} \qquad (5)$$

$$x(r_i, e_j) \in \{0, 1\} \qquad {\scriptstyle \forall r_i \in R, e_j \in E} \qquad (6)$$

The objective function minimizes aggregate delay between routers and their assigned DEs. Constraint (2) ensures that each router in R is assigned, (3) ensures that the DE workload capacities are not violated, and (5) imposes the contiguity requirement.

The load balancing constraint (4) is weighted by a parameter, Δ, which controls the maximum deviation of a DE's normalized workload from the average normalized workload for all DEs. Setting $\Delta = 1$ would force workloads of all DEs to be exactly equal to the average normalized workload, or in other words each DE will have the same fractional utilization of its capacity as all others. In case of homogeneous DE capacities this translates to an equal workload for all DEs. On the other hand, $\Delta > 1$ allows the normalized workload of at least one DE to be higher than the average by $(\Delta - 1) * 100$ percentage.

The value of Δ also dictates the trade-off between the objectives of minimum aggregate delay and load balancing as it changes the feasible set of solutions. A large value of Δ optimizes a solution for the objective of minimizing aggregate delay, while a tighter constraint will show significant trade-off in favor of load balancing. The addition of a hard constraint for load balancing comes at the cost of reduced feasibility where optimal solutions could be infeasible because of a choice of Δ which is too low. This situation is likely to arise in tightly constrained problems especially in the event of reduced capacity as a result of DE failures. However, the dependence of (4) on the average normalized workload ensures that the formulation dynamically adapts to failures, as a DE failure lowers the total available capacity thereby increasing right hand side of the constraint. This will result in higher workload shares for the remaining functional DEs to accommodate the orphaned routers. If the total capacity of the remaining DEs is less than the workload offered by the Data plane, no feasible solution will exist for the problem.

Our approach is different from the traditional load balancing method of minimizing the maximum load, and provides better control to a network operator while ensuring robust and efficient operation of the Decision plane. The sub-problem with only the minimum delay objective and (2), (3) and (6) is commonly referred to as Terminal Assignment Problem [11].

B. Two-phase Router Assignment Algorithm

We construct a two-phase exact algorithm to solve the optimization problem. The first phase of the algorithm constructs an ordering of routers, S, where S is the sorted order of minimum delay assignments for each router, and greedily assigns routers in the order of S to their closest (min-delay) feasible DEs, if such assignments are possible. To meet the contiguity constraint (5), a router r_i's assignment is made to the closest DE e_j if $d(r_i, e_j)$ is strictly less than the delay between r_i and any other DE and e_j has slack capacity. On the other hand, if there are other DEs at same delay from r_i as e_j, r_i is assigned to a feasible DE that has an existing assignment in $N(r_i)$. Otherwise, r_i is kept unassigned.

The goal of the first phase of algorithm is to make all feasible lowest-cost assignments that can be made without changing any previously made assignments. This phase constructs an optimal solution for the assigned routers. Any routers that remain unassigned after the first phase are assigned by the second phase using a branch exchange algorithm that iteratively accommodates previously unassigned routers, while maintaining feasibility of the solution. Our solution is $O(m^2 n)$ in the worst case, and finds optimal solution to the assignment problem if it exists.

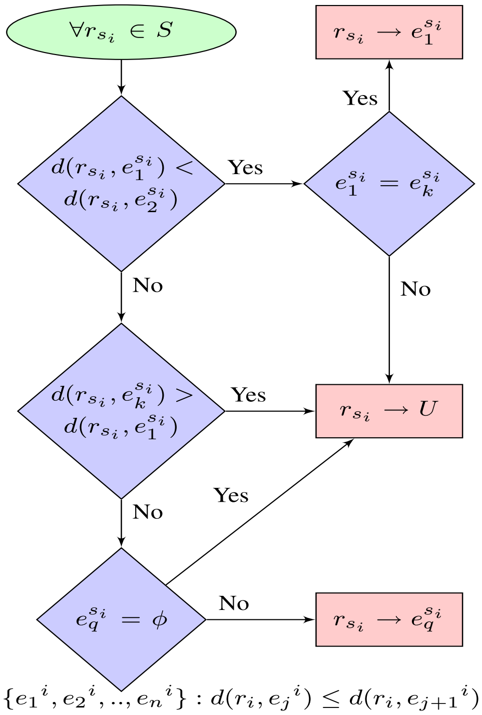
1) Greedy Phase: We utilize a greedy heuristic to assign routers to DEs while maintaining the feasibility of solution. Since, by definition, a greedy approach does not make any changes to its local decisions, the order in which decisions are taken becomes important. Our approach considers routers in the order of lowest assignment costs for each router. Assignments are made only with a feasible min-delay DE, where feasibility is determined by the constraints given in §IV-A. Fig. 3 describes the definitions and operation of this phase.

Lemma 1: Let $x(r_{s_i}, e_k^{s_i})$ be an assignment made in the greedy phase. By construction, $d(r_{s_i}, e_k^{s_i}) \leq d(r_{s_i}, e_j^{s_i})\ \forall e_j^{s_i} \in E^{s_i}$ i.e. $e_k^{s_i}$ must be the minimum cost feasible assignment for r_{s_i}.

The algorithm explicitly checks a potential assignment against the capacity (3) and load balancing (4) constraints, while implicitly meeting the contiguity constraint (5) according to the following Lemma:

Lemma 2 (Greedy Phase meets (5)*):* Since router assignments are done strictly in the order of min-delay, it suffices to show that routers assigned in this order will meet the contiguity constraint. We prove this Lemma by induction on the assignment of a router r_{s_i}: If $A(e_1^{s_i}) = \phi$, the Lemma trivially holds as r_{s_i} must be directly connected with $e_1^{s_i}$ by Lemma 1. For the case of $A(e_1^{s_i}) \neq \phi$, we assume that Lemma holds for $i - 1$ assignments and r_{s_i} is the i^{th} assignment that violates the Lemma, implying $\exists r_a \notin A(e_k^{s_i})\ \forall r_a \in N(r_{s_i})$

Conditioning on r_a, we observe that there must be a path from r_{s_i} to $e_k^{s_i}$ which passes through r_a. Hence, $d(r_{s_i}, e_k^{s_i}) = d(r_{s_i}, r_a) + d(r_a, e_k^{s_i})$ which implies $d(r_a, e_k^{s_i}) < d(r_{s_i}, e_k^{s_i})$. Therefore, r_a must have been picked by the algorithm before r_{s_i} and since $e_k^{s_i}$ is a feasible choice for r_{s_i} it must have been a feasible choice for r_a. This implies r_a is assigned to an arbitrary DE e_1^a where $e_1^a \neq e_k^{s_i}$ and $d(r_a, e_1^a) < d(r_a, e_k^{s_i})$. By substitution, it can be seen that this results in $d(r_{s_i}, e_1^a) <$


$\forall r_{s_i} \in S$
$d(r_{s_i}, e_1^{s_i}) < d(r_{s_i}, e_2^{s_i})$
Yes
$e_1^{s_i} = e_k^{s_i}$
Yes
$r_{s_i} \rightarrow e_1^{s_i}$
No
No
$d(r_{s_i}, e_k^{s_i}) > d(r_{s_i}, e_1^{s_i})$
Yes
$r_{s_i} \rightarrow U$
No
Yes
$e_q^{s_i} = \phi$
No
$r_{s_i} \rightarrow e_q^{s_i}$

$E^i = \{{e_1}^i, {e_2}^i, .., {e_n}^i\} : d(r_i, {e_j}^i) \leq d(r_i, {e_{j+1}}^i)$

$S = \{r_{s_1}, r_{s_2}, .., r_{s_m}\} : d(r_{s_i}, {e_1}^{s_i}) \leq d(r_{s_{i+1}}, {e_1}^{s_{i+1}})$

$U = \{\text{Set of unassigned routers}\}$

$k =$ Index of the first feasible DE in E^i

$e_q^{s_i} \in E^{s_i} \quad k \leq q < n :$

$\exists r_a \in N(r_{s_i}), A(e_q^{s_i})$

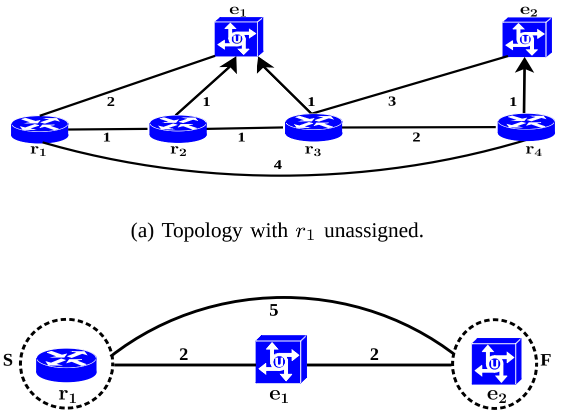
$d(r_{s_i}, e_q^{s_i}) = d(r_{s_i}, e_1^{s_i})$

Fig. 3. Greedy Phase Algorithm

$d(r_{s_i}, e_k^{s_i})$, thus violating Lemma 1. Therefore, i^{th} assignment must be valid.

2) Exchange Phase: The greedy phase makes all the feasible min-cost router assignments that can be made without changing any existing assignment. Consequently, assignment of an unassigned router after the greedy phase's completion may involve a trade-off between sub-optimal assignment to available DEs or reassignment/exchange of already assigned routers to allow a lower cost assignment. Therefore, in order to ensure optimality of the solution, the assignment mechanism must be able to find the lowest-cost set of exchanges that allow the assignment of an unassigned router. This mechanism is provided by the exchange phase, which utilizes a branch-exchange algorithm, similar in design to the method described in [11], to construct an auxiliary graph of the network and uses shortest path algorithm for computing lowest-cost assignments.

In simple terms, auxiliary graph represents the feasible combinations of router assignment exchanges between DEs, weighed by the cost of such exchanges. The min-cost path through the graph represents the min-delay assignment for a previously unassigned router. Therefore, edges of the graph represent possible feasible exchanges (and new assignments) between DEs which, themselves, are represented by the graph's vertices. Similar to the greedy phase, feasibility of any exchange or new assignment depends on conformance to the constraints presented in §IV-A. Auxiliary graph is constructed



(a) Topology with r_1 unassigned.

(b) Auxiliary graph where $(S, e_1) = e_1 \leftarrow r_1$ and $(e_1, F) = e_2 \leftarrow r_3$

Fig. 4. Operation of the exchange phase on a network example where $\Delta = 1$ and edges are annotated with delay values. The min-cost assignment is along $(S, e_1), (e_1, F)$

according to the following rules:

- There are two special vertices S and F that represent the source and destination vertices for the shortest path computation. The shortest path from S to F, at each iteration of exchange phase, provides the lowest cost assignment of one unassigned router.
- There are additional vertices, $Y = Y_1, Y_2, .., Y_k$, each corresponding to a fully loaded DE.
- There is an edge (S, Y_k) corresponding to potential assignment of an unassigned router $Y_k \leftarrow r_i : \exists r_a \in A(Y_k),\ r_a \in N(r_i)$ with an edge weight $d(r_i, Y_k)$.
- There is an edge (Y_k, Y_l) corresponding to a router r_i at the border of Y_k and Y_l's areas, such that $x(r_i, Y_k) = 1$, $\exists r_a \in A(Y_l), r_a \in N(r_i)$ and the weight $d(r_i, Y_l) - d(r_i, Y_k)$ is positive.
- There is an edge (Y_k, F) corresponding to a router r_i's feasible re-assignment from Y_k to a DE e_j with slack capacity. The weight of this edge is $d(r_i, e_j) - d(r_i, Y_k)$.
- There is an edge (S, F) with weight $d(r_i, e_j)$ for $e_j \leftarrow r_i$.

Lemma 3 (The auxilary graph has no negative cycles): There can not be any negative cycles involving S and F vertices, and so it only remains to be shown that the vertices in Y do not have any negative cycles between them. We observe that only edges with positive weights are allowed between vertices in Y, and since a negative cycle implies edges with negative weights, the Lemma is proven by construction.

Dijkstra's shortest path algorithm is used to compute the shortest path from S to F on the directed auxiliary graph. Lemma 3 establishes that Dijkstra's algorithm, which can only be used in graphs with no negative cycles, is applicable to the auxiliary graph. This shortest path represents the minimum cost set of exchanges that are needed to assign a previously unassigned router. The auxiliary graph is updated after the assignment and the process repeated until all routers have been assigned. Fig. 4(a) shows the operation of the exchange phase for a simple network example and Fig. 4(b) shows the construction of its auxiliary graph.

3) Complexity: The greedy phase of the algorithm is $O(m)$. The exchange phase's complexity is dependent on the shortest path computation, with worst case complexity of $O(n^2)$. The exchange phase calls Dijkstra's algorithm for each unassigned router, resulting in an overall worst case complexity of $O(mn^2)$. In reality, the greedy phase assigns most of the routers, and the few unassigned routers in tightly-constrained DE failure scenarios each require one iteration of the exchange phase. This results in average-case complexity of $\Theta(m + kn^2)$, where $k \ll m$. Also, since the number of routers in a network are expected to be much higher than the number of DEs, i.e. $m \gg n$, complexity of the scheme is dominated by the complexity of greedy phase, resulting in very fast run-times e.g. less than $3.5s$ on average in a network with $(m, n, \Delta) = (1500, 10, 1.0)$ as described in §VI.

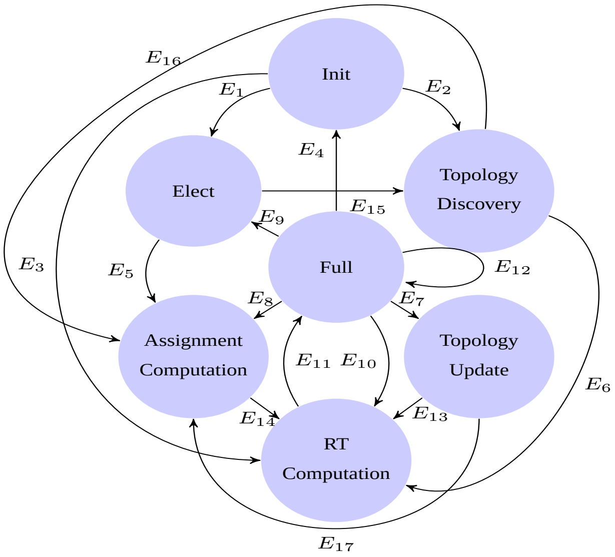
V. DPP Protocol for Decision Plane Operation

In this section we discuss the design of an experimental protocol for the operation of logically centralized Decision plane using the router assignment algorithm. A discussion of the main functional requirements of DPP protocol is presented, followed by a description of the protocol structure and states, and finally we discuss how the protocol interacts with other layers of the 4D architecture.

A. Functional Requirements

The protocol operating at the Decision layer is responsible for management of DEs in providing a uniform network-wide Decision plane. To effectively meet the design goals specified in §I, the design needs to conform to the following basic functional requirements:

- Robustness to multiple failures in the Decision and Data planes must be insured. This implies a design that incorporates redundant control logic and storage of network state.
- Any pre-configuration of protocol parameters should be minimized and the protocol must be able to operate without constant human intervention.
- Protocol must be easily extensible and evolvable to include additional functionalities.
- To improve scalability of the Decision plane, the protocol must distinguish between events which have network-wide significance vs. events which have their impact limited within a local DE's control boundaries. For example, failure of a redundant link totally contained within a local area may not have AS-wide significance, while failure of a backbone link connecting two different areas might require re-computation of routing matrices at multiple DEs to redirect traffic away from the affected link.
- The protocol must be able to deal with synchronization issues expected in the control of a large geographically-dispersed AS.



Event	*Description*
E_1	Network Bootstrap
E_2	Addition of a new DE in the network
E_3	Reception of topology and assignment from leader
E_4	Reboot
E_5	Only if not in network bootstrap
E_6	Reception of assignment from leader
E_7	Local area event
E_8	DE failure or router addition. (Leader only)
E_9	Leader failure
E_{10}	Reception of new assignment or reachability update
E_{11}	Send RTs and reachability update
E_{12}	Stable network
E_{13}	Reception of new assignment or intra-area event
E_{14}	Send the assignment to other DEs in the network
E_{15}	Network Bootstrap
E_{16}	Only in the case of leader DE
E_{17}	Inter-area event (Leader only)

Fig. 5. State transition diagram for the Decision Plane Protocol

These requirements are not meant to be exhaustive but to serve as a guideline for the protocol design.

B. Protocol Design

The functional requirements of the previous section provide a basis for the design of DPP protocol where we incorporate the following salient design features:

Leader Election: Router assignment algorithm is computed only by the DE which has been chosen to act as leader. We utilize a simple leader election protocol based on unique pre-configured DE identifiers. The leader election protocol is used at network bootstrap, after the setup of control paths between DEs, and leader's failure event. This mechanism fulfills the design requirements in several ways. Firstly, it does not require any pre-configuration on part of network operator beyond the DE identifiers. Secondly, it avoids the potential assignment conflicts that could arise due to asynchronous computation of assignments by DEs. Finally, it allows a robust design as failure of any particular machine does not jeopardize the network operation.

Network State and Logic: The network state, consisting of the topology information of Data plane and routes

Construct	Function
`get_topo()`	Request network topology discovery from the 4D Discovery plane.
`send_RT()`	Send a new RT to the specified router using the 4D Dissemination plane.
`push_event()`	Used by the 4D Dissemination plane to signal an event in a DE's area

TABLE I
APIS USED FOR INTER-LAYER COMMUNICATION

advertised by DEs, is replicated across the Decision plane. The route advertisements, in the form of DE-DE messages, provide reachability information about a DE's area. Frequent collection of topology information from the lower layers of the architecture is avoided as it is a costly process in terms of overhead and delay. This is because the abstraction of area boundaries does not extend to any lower layers and a request from the Decision plane for collecting topology information encompasses the entire network topology. Therefore, we limit topology discovery to the cases of network bootstrap and new DE addition only. In other cases, e.g. when a DE is restarted after a failure, topology discovery is not required as it had been done previously and the persistent network state can be acquired from the current leader along with router assignments.

We make a distinction between events at Data plane by categorizing them into, 1., Inter-area events, i.e. those affecting links and routers between Decision plane areas and, 2., Intra-area events, which are contained within a DE's area. Only the former category of events require re-computation of router assignments and each DE is responsible for the computation and dissemination of Routing Tables (RT) for its assigned routers.

Interaction with Other 4D Layers DPP is designed to require only a small set of APIs from the underlying layers of the 4D architecture, as listed in table I. This mechanism is selected with the aim of improving extensibility of the architecture, allowing this basic set of APIs to be re-used in any additional control features beyond shortest-paths routing. The implementation of these APIs in the lower architectural layers is not explored in this work.

C. Protocol States

A DE is transitioned through several states from initialization to full operation and undergoes further state changes in response to network events. Fig. 5(a) illustrates the state machine of the DPP protocol where we utilize the following states to describe its operation:

Init or initialization state follows immediately after boot-up. Secure channels for the exchange of control messages are immediately established with each of DE's neighbors in the fully-connected Decision plane. If there are no previously initialized neighbors, all DEs are transitioned through the leader election protocol. Otherwise, the current leader checks a newly booted DE's identifier to find out if it was previously initialized.

Elect state is used when there is no leader DE in the network, which will be the case at network bootstrap, or in case of leader's failure. Each DE in the network is pre-configured with a unique integer identifier. The DEs exchange their identifiers to elect the one associated with the lowest identifier as leader.

Topology Discovery In this state, network topology information is requested from the 4D Discovery layer using the `get_topo()` construct. The topology is in the form of a weighted graph where vertices indicate routers and edges specify physical adjacencies, which are weighted by propagation delay of the links. The topology information is exchanged between DEs to ensure full replication of network state across the Decision plane.

Router Assignment The leader DE transitions into this state in the event of a DE failure, failure at inter-area links, or an addition of a new router.

Routing Table Computation is done by each DE for the routers in its area whenever it receives a new assignment from leader DE, in case of intra-area events, and when it receives new reachability information from another DE. The completion of routing table computation is immediately followed by an update of each router's routing table using the `send_RT()` construct to the 4D Dissemination plane, and an update of reachability information to other DEs if the new computation results in changes to the routes available to their areas.

Topology Update is a result of an event in a DE's area. It requires sending topology update to other DEs in Decision plane in order to synchronize the network state. A `push_event()` construct allows 4D Dissemination plane to signal such events to the Decision plane.

Full DE in this state indicates a fully initialized Decision plane. This state would be maintained in normal operation.

VI. NUMERICAL EVALUATION

In this section we provide results of our evaluation of the assignment algorithm on real-world and a variety of artificially generated topologies.

The first set consists of the ISP backbone topologies collected by Rocketfuel project [12]. The second set are artificial two-tiered hierarchical topologies generated by BRITE [13] using the GLP model [14]. GLP model along with BRITE has been reported to generate ISP-like topologies [15], which we use to model a large-sized ISP topology consisting of 1500 routers and 15 DEs. Our evaluation was focused on determination of the following characteristics:

1) Reassignment of non-orphaned routers: The accommodation of routers orphaned as a result of a DE failure, may necessitate re-assignment of non-orphaned routers from other DEs to balance the load among the surviving DEs. A large percentage of such reassignments could have an adverse effect on the Decision plane performance and it is desirable to reduce such router churn. We measure this as a percentage of non-orphaned routers undergoing re-assignment out of the total number of routers in the network.
2) Computation time: Each failure in the Decision plane triggers the re-computation of the router assignments.

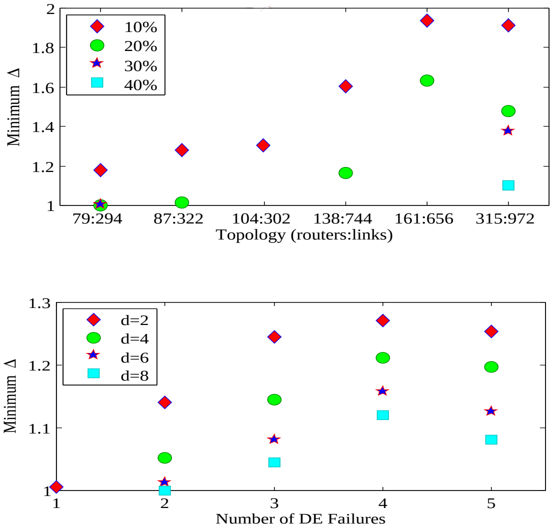


Fig. 6. Trade-off between load balancing and percentage of non-orphaned router re-assignment. Plots show the minimum value of Δ needed to limit the re-assignments below a given percentage. Top: (a) Rocketfuel backbone topologies, Bottom: (b) BRITE topologies of $m = 1500$ with max. 5% re-assignments

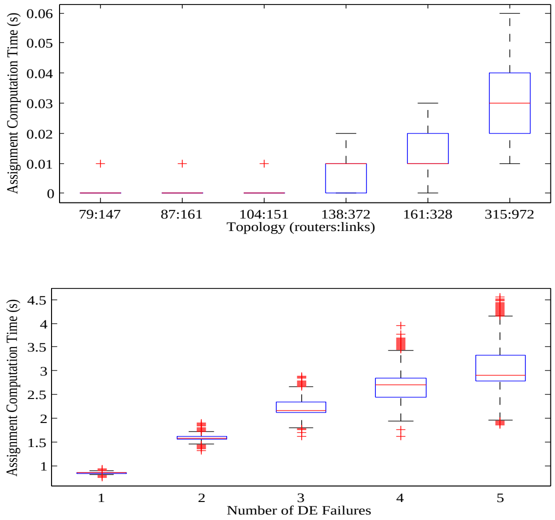


Fig. 7. Box Plot of the computation time for router re-assignment with $\Delta = 1$. The box shows the first and third quartile along with the median. Whiskers show the min. and max. values, while the outliers are plotted as "+". Top: (a) Rocketfuel backbone topologies, Bottom: (b) BRITE topologies with $m = 1500$

We measured the time taken for each run of the assignment algorithm on a 64 bit 3.6 Ghz machine.

In each topology, we determine the best positioning of a set of DEs based on the discussion in [8]. Results were obtained by removing all combinations of "failed" DEs from the original set. Maximum number of DEs (n_{max}) was limited to 15 in BRITE and 10 in Rocketfuel sets. The minimum number of DEs n_{min} was constant at 5 in both sets, which was found to be sufficient in attaining near-optimal convergence delays [8]. The capacities of individual DEs were assumed to be a non-limiting factor and, in the case of BRITE set, our experiments were repeated for different degree distributions (d) of Decision plane areas.

Fig. 6(a) shows non-orphaned router reassignment for the case of Rocketfuel backbone topologies, where we present results by bounding the maximum percentage of router reassignments in a network and presenting the minimum value of Δ that is needed to ensure that reassignment rate remains below the bound. We observe that even in this very limiting case of backbone topologies, the rate of reassignment falls off rapidly with an increase in Δ and relatively small values of Δ are sufficient in achieving tight bounds on router reassignment. In the case of BRITE topologies, we observe even better performance as full topological information is available. Fig. 6(b) shows results for the case of BRITE topologies where we report the observed minimum values of Δ required in bounding maximum reassignments to 5% for different area degrees.

The computation time required to run each iteration of the algorithm is plotted in Fig. 7 for both sets of topologies, with a worst-case DE capacity constraint of $\Delta = 1.0$. The plot shows that even in case of very large network topologies and worst-case constraints on load-balancing router assignment algorithm converges to a solution within very reasonable times.

VII. Simulation Results

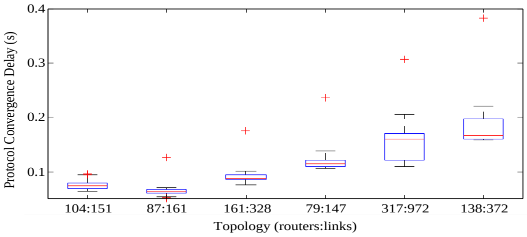
We analyzed the convergence performance of the DPP protocol with simulations on Rocketfuel topologies used in the previous section, using ns-2 simulator [16] where we created new modules to implement the functionality of 4D Decision plane. We collected results on the convergence delay in cases of network bootstrap and DE failures.

The convergence delays are computed by randomly forcing the failure of a DE and measuring the time until all routers in the network receive re-computed routing tables. This convergence delay includes, 1., delay at the Decision plane between the time a failure actually occurs and when it is detected by the functional DEs, 2., computation time of router assignment algorithm, 3., reception of new assignments by the DEs, 4., new routing table computation, and 5., reception of new routing tables at each router. The Decision plane failures are detected by a DE keep-alive timers which expire when no keep-alive message is received by a neighboring DE within a time period equal to the maximum delay between DEs. We utilized results obtained in the previous section for routing assignment computation time while RT computation time was kept constant at 1ms. Simulation were repeated for the range of DE failure combinations with $n_{\text{max}} = 10,\ n_{\text{min}} = 3$.

Table 8(b) shows convergence and maximum network delays in the case of network bootstrap. Box plot of the convergence delays is shown in Fig. 8(a). The results show that DPP protocol achieves sub-second convergence delays even in largest of the simulated topologies.

VIII. Related Work

Several recent studies have embraced centralization of network logic as a way of overcoming management complexity



Topology	***Max. Network Delay***	***Bootstrap Delay***
(routers:links)	(ms)	(ms)
104:151	28	95.13
87:161	35	126.35
161:328	47	175.12
79:147	72	235.3
317:972	86	306.4
138:372	97	383.2

Fig. 8. Simulation results of protocol convergence delay for Rocketfuel topologies with $n_{\max} = 10$. Top: (a) Box Plot of the protocol convergence delay with $\Delta = 1$. The box shows the first and third quartile along with the median. Whiskers show the min. and max. values, while the outliers are plotted as "+". Bottom: (b) Bootstrap convergence delays for Rocketfuel topologies

or providing new services that are presently difficult to implement. Greenberg et al. [2] provide a comprehensive survey of the issues in network control and management, and propose the architectural vision embraced and extended in this paper.

Centralized control has been explored in BGP design where RCP [1], [17] was proposed as a logically centralized point for computing BGP routes and improving the scalability of large networks. However, RCP is limited to BGP route computation and does not extend to Interior Gateway Protocol (IGP) routes.

Recently, CONMAN [18] utilized the concept of management plane and centralization in the design and operation of "network managers" that are used to manage the protocols running on individual routers.

Several efforts in open router design [10], [19] have also advocated migration of control functions away from routers to reduce their complexity, where they utilize "control elements" for the implementation of distributed network algorithms, and design protocols to enable communication between different network elements. In contrast, our work uses 4D's approach of network-wide decision making and presents a robust and scalable design for the Decision plane that is not limited to the implementation of current distributed algorithms.

IX. Conclusion

We presented the design of a clean-slate control and management plane to simplify the management complexity in large enterprise and ISP networks. Within the architectural framework of 4D architecture, the proposed Decision plane is designed to meet its stated goals of achieving high scalability and robustness, while minimizing the response time to any event.

Our work included a novel method of adaptive assignment of routers to the logically centralized Decision Elements (DE) and a protocol that allows distributed operation of the DEs while ensuring replication and synchronization of network state across the entire Decision plane. The evaluation of our protocol and algorithm through different mechanisms and models supports the adherence of our design to its goals, and the feasibility and benefit of using a logically centralized Decision plane.

The main avenues of related future research include the interoperable design of lower 4D layers, extension of management functions e.g. to include provision for system maintainence, and detailed specification and analysis of DPP.

References

[1] N. Feamster *et al.*, "The case for separating routing from routers," in *ACM SIGCOMM FDNA Workshop*, 2004.

[2] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *SIGCOMM CCR.*, vol. 35, no. 5, 2005.

[3] D. A. Maltz, G. Xie, J. Zhan, and H. Zhang, "Routing design in operational networks: A look from the inside," in *in Proc. ACM SIGCOMM*. ACM Press, 2004.

[4] A. Wool, "A quantitative study of firewall configuration errors," *IEEE Computer*, vol. 37, no. 6, 2004.

[5] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, 2002.

[6] A. Greenberg *et al.*, "Refactoring network control and management: A case for the 4D architecture," Carnegie Mellon University, Tech. Rep. CMU-CS-05-117, Sept 2005.

[7] H. Yan *et al.*, "Tesseract: A 4D network control plane." in *USENIX NSDI*, 2007.

[8] H. Iqbal and T. Znati, "Distributed control plane for 4D architecture," in *Globecom 2007*. IEEE, 2007.

[9] I. T. Union, "ITU-T recommendation Q.700: Introduction to CCITT Signalling System No. 7," 1993.

[10] T. V. Lakshman *et al.*, "The SoftRouter architecture," in *HotNets-III*, 2004.

[11] A. Kershenbaum, "Telecomm. network design algorithms," 1993.

[12] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *In ACM SIGCOMM IM Workshop*, 2002.

[13] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," 2001.

[14] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," in *in Proc. IEEE INFOCOM*, 2002.

[15] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "On realistic network topologies for simulation," in *SIGCOMM MoMeTools Workshop*, 2003.

[16] *The Network Simulator - ns-2*, Information Sciences Institute.

[17] M. Caesar *et al.*, "Design and implementation of a routing control platform," in *USENIX NSDI*, 2005.

[18] H. Ballani and P. Francis, "CONMan: a step towards network manageability," *SIGCOMM CCR*, vol. 37, no. 4, 2007.

[19] L. Yang, R. D. T. Anderson, and R. Gopal, "RFC 3746 Forwarding and Control Element Separation Framework," 2004.