DA VINCI: A Tool to Improve VoIP Call Routing Configurations

Kunwadee Sripanidkulchai, Shu Tao, and Zon-Yin Shae
IBM T.J. Watson Research Center

Abstract—As IP telephony becomes a critical part of the enterprise network infrastructure, there is an increasing need for tools to help manage VoIP systems. In this paper, we introduce DA VINCI, a tool to manage VoIP call routing — the core of VoIP configurations that determines how calls are routed inside the enterprise or between the enterprise and external networks. We first establish the complexity involved in call routing design and configuration by studying characteristics of nine large enterprise VoIP networks. We then develop methodologies to improve and simplify call routing configurations by (i) identifying common design patterns, i.e., shared resource routing and local resource routing, so that administrators can better understand and reuse existing call routing designs, and (ii) removing unused and superfluous configurations from dial plan designs while retaining the correctness of operations. All of our techniques are incorporated in the DA VINCI tool that has been deployed in more than ten enterprise networks to help administrators streamline VoIP maintenance through extraction, analysis, simulation, and configuration of call routing.

I. INTRODUCTION

In recent years, IP telephony has become a critical part of the enterprise network infrastructure. Compared to the traditional public switched telephone network (PSTN), VoIP offers lower cost and greater flexibility in implementing call routing policies. For example, for an enterprise with geographically dispersed sites connected by IP networks, calls from the enterprise to the external PSTN can be routed through different gateways to save cost. Such flexibility, however, also gives rise to greater challenges in designing, configuring, and managing call routing, especially for large enterprises.

VoIP call routing is controlled by a dial plan, which consists of a set of routing rules configured in the call routing server, e.g., Cisco CallManager. Routing rules are implemented by defining and piecing together different functional components such as Calling Search Spaces, Partitions, Route Lists, Route Groups, etc., in Cisco’s terminology. Although VoIP vendors provide general guidelines for configuring these functional components, the overall design and management of the dial plan remains a complex and challenging task for network administrators. In practice, dial plan design is pervasively manual and ad-hoc. As a result, the VoIP deployment process is time-consuming typically ranging from 6 to 18 months, and the resulting dial plan is often error-prone.

In order to streamline VoIP design and configuration, we make four important contributions in this paper. Our first contribution is to establish the complexity of call routing design based on configuration data gathered from nine operational VoIP networks. To our knowledge, this is the first study that reports on enterprise VoIP configuration and routing.

Our second contribution is to identify common call routing design principles that can be used as templates to simplify the configuration of new dial plans. Despite the fact that each enterprise has its unique requirements, we establish that similar VoIP routing design principles are common across enterprises. Specifically, shared resource routing often used for cost reduction reasons and local resource routing often used for implementing geography-related policies are two common design patterns used in all studied enterprises.

Our third contribution is to improve the maintenance of existing dial plans. Like any network or system configuration, a dial plan is frequently updated as the underlying VoIP infrastructure or the call routing policies change. Such changes may result in erroneous or left-over configurations. To address this challenge, we develop methodologies to improve dial plans through the identification and removal of unused configuration components and superfluous configurations.

We implement all of our contributions in a tool that can facilitate the design, deployment, and maintenance of dial plans called DA VINCI (Dial plan Analysis, VIsualization, aNd Configuration Integrated toolset). To date, this tool has been used in more than ten VoIP networks and has shown great potential in dramatically shortening deployment and debugging time, and reducing configuration errors.

The remainder of the paper is organized as follows: Section II briefly introduces the background on VoIP, call routing and dial plans. Section III establishes the complexity of VoIP configurations based on configurations from nine enterprise VoIP networks. In Section IV, we describe the implementation of the DA VINCI tool to help stream-line VoIP deployment and operations. We propose methodologies for simplifying and improving dial plans by leveraging common design patterns in Section V and removing unused and superfluous configurations in Section VI. Finally, Section VII discusses related work and Section VIII concludes the paper.

II. BACKGROUND

In this section, we review the background on VoIP network infrastructures. We then discuss call routing configurations (i.e., dial plans) and our graph-based model of dial plans.

Fig. 1. A simplified enterprise VoIP infrastructure.
A. VoIP Infrastructure

Fig. 1 depicts a simplified example of an enterprise VoIP infrastructure. In this example, the enterprise has two sites at different locations, site A and B. The enterprise has multiple VoIP clusters, each corresponding to the signaling proxy or call routing server (i.e., Cisco CallManager) at each site that implements its own call routing logic inside its dial plan. Both clusters are connected to the enterprise IP network.

In each cluster, phone devices such as hardware or software-based phones can be used to initiate or receive calls. Phones first register themselves with a signaling proxy such as Cisco CallManager. Using signaling protocols, such as SIP [1], H.323 [2], or the Skinny protocol [3], the signaling proxy relays call initiation and termination messages between the callers and callees. Once a call is successfully initiated a media path will be established between the end-devices to transport the voice data. If both caller and callee are using IP phones inside the enterprise, then the media path could be a direct IP path. Otherwise, the media path will traverse the voice gateway so that the media data can be translated and carried across the IP network and the PSTN. The role of call routing is to determine how the call should be routed and which resources the call ought to use in the infrastructure while considering factors such as fault tolerance, availability, load balancing, or cost-effectiveness.

B. Graph-Based Dial Plan Model

Call routing logic is controlled and implemented in the call routing server (e.g., Cisco CallManager) as a configuration called the dial plan. The dial plan matches a call to pre-defined patterns (i.e., called phone number) based on where the call originated from, and then maps it to a destination routing resource such as a voice gateway, trunk, phone, etc. The information used for this mapping includes calling number, called number, and the originating device. For instance, two calls to the same destination number but originating from the PBX and from an IP phone may be routed differently.

The language used for defining dial plan is proprietary. Many VoIP vendors, e.g., Cisco [4] and Avaya [5], and open-source alternatives, e.g., Asterisk [6], use their own terminology to define their dial plans. Nevertheless, their dial plans all share similar concepts. In this study, we use Cisco’s dial plan terminology. For future work, the analysis and the derived methodologies may be generalized to dial plans on other platforms.

Cisco defines the following functional components that can be composed together to specify a dial plan. Each component has a set of configurable parameters that implement various functions to support call routing [7].

- **Routing Devices**: Gateways or trunks that route calls from a VoIP network to another network (IP or non-IP), or vice versa. Examples of such devices include Cisco gateway, H.323 gateway, SIP trunk, etc.
- **Route Group**: A group of routing devices that are ordered by the administrator to prioritize or load-balance gateways and ports for outgoing gateway/trunk selection.
- **Calling Search Space (CSS)**: A CSS is also used to logically divide up the dial plan and contains an ordered list of partitions. Directory numbers, phone devices and routing devices are each assigned calling search spaces. When a call is initiated, the call routing process looks for the CSS assigned to the calling number or device. Then, the called number is matched to the patterns in the partitions assigned to the CSS, and so on and so forth as described next.

Dial plans are constructed by piecing together the above components to implement a range of complex routing logic. We model the dial plan as a directed graph in which nodes represent the functional components and edges represent the association between the components. For example, if CSS \( C_i \) contains partition \( P_j \), the graph will have an edge \( C_i \rightarrow P_j \); if in partition \( P_j \), a route pattern is defined so that the matching calls will be routed to route list \( L_k \), then the graph will have an edge \( P_j \rightarrow L_k \). Table I summarizes all possible edges between different types of nodes in the dial plan graph.

![Figure 2(a)](image)

**Figure 2(a)** depicts the traversal for an outgoing call where a directory number (1111) which is an extension inside the enterprise makes a call to an external phone number such as 911. The routing starts with the calling number which is configured in Partition1. Partition1 has at least one associated CSS (CSS2). CSS2 looks in all of its associated partitions for route patterns or translation patterns that match the called number. If a match is found with a route pattern, the call is then processed by the route list, route group, and finally sent off to a routing device to finish call routing.

Using the graph model, determining the call routing path is a graph traversal problem. Consider a call containing calling and called number information. As a call traverses the graph, each node on the path (i.e., the above functional components) may process and transform the calling and called numbers until the traversal completes. The starting point of the traversal is determined by the caller’s CSS (i.e., CSS for the calling number or calling device for an outgoing call or CSS for the routing device that the call arrived at for an incoming call). The resulting path through the graph is determined by the called number.

<table>
<thead>
<tr>
<th>Source node type</th>
<th>Sink node type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>CSS, Route List, Routing Device</td>
</tr>
<tr>
<td>Route List</td>
<td>Route Group</td>
</tr>
<tr>
<td>Route Group</td>
<td>Routing Device</td>
</tr>
<tr>
<td>CSS</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE I**

**Edges Between Different Types of Nodes in a Dial Plan Graph.**

- **Route List**: A prioritized list of route groups. Each route list is associated with one or more route patterns (i.e., called phone numbers defined in partitions) and determines the order of route groups that will route the call.
- **Partition**: Partitions are used to divide the dial plan into logical subsets based on organization, location, call permissions, or call type. A partition may contain a list of directory numbers (phone number extensions used inside the enterprise), route patterns (regular expressions that match with called phone numbers and map to route lists), and translation patterns (defined to manipulate the caller or callee’s numbers as part of the routing process).
- **Calling Search Space (CSS)**: A CSS is also used to logically divide up the dial plan and contains an ordered list of partitions. Directory numbers, phone devices and routing devices are each assigned calling search spaces. When a call is initiated, the call routing process looks for the CSS assigned to the calling number or device. Then, the called number is matched to the patterns in the partitions assigned to the CSS, and so on and so forth as described next.
Incoming call path from outside the enterprise to directory number 2222.

CA and CO1
NY2 is a test/pilot cluster, both serving two
CA has a little over 100 nodes.
EU cluster serves an entire
NY2) and one
NY1, CO1
NY1 is a production

is found with a translation pattern, the calling and called numbers may be manipulated and then sent to a CSS to begin another round of call routing processing. The process may run multiple rounds through the graph, each round traversing different translation patterns until finally a route pattern is traversed and call routing finishes at a routing device.

Incoming calls are processed in a similar fashion. Figure 2(b) depicts an incoming call to the enterprise extension 2222. The call starts from the PSTN gateway (routing device 2) which is associated with a CSS (CSS3). In the partitions associated with this CSS3, the called number will then be matched against the configured route patterns, translation patterns, or directory numbers. When a match is found with the called directory number, the call routing is finished. However, if a match is found with a route pattern or translation pattern, it will be processed by its downstream nodes in the dial plan graph. This continues until either the called directory number 2222 is found or the call is forwarded to a routing device.

III. COMPLEXITY OF DIAL PLAN CONFIGURATIONS

In this section, we establish the complexity involved in call routing design and configuration by studying characteristics of nine large multi-national enterprise VoIP networks between Dec. 2008 and Apr. 2009. The data is collected by running the dial plan extraction tool described later in Section IV.

A. VoIP Deployment Characteristics

We extracted a total of nine dial plans from three multinational enterprises listed in Table II. Each dial plan is from a cluster as defined in the enterprise. The largest enterprise in our study, CO1, has a large VoIP infrastructure spanning multiple continents. In this study, we sample the dial plans for five of its VoIP clusters. Four of these clusters are located in North America (CO1_CA, CO1_CT, CO1_NY1, CO1_NY2) and one is located in Europe (CO1_IT). CO1_NY1 is a production cluster and CO1_NY2 is a test/pilot cluster, both serving two proximate sites and sharing the same number space. Each of the other 2 clusters in North America, CO1_CA and CO1_CT, serves an individual site. The CO1_EU cluster serves an entire country in Europe with roughly 20 sites.

The second enterprise, CO2, has presence in three continents, Asia, Europe and North America, each of which is designed as a cluster and served by a dial plan. There are about 10 sites dispersed Europe and North America, respectively, and a couple of sites in Asia. The third enterprise, CO3, also has presence in the same three continents as CO2. However, we extracted only one of its dial plans for the cluster in North America that serves about 10 different sites. Note that each dial plan can cover an individual site, multiple sites, a country, or even multiple countries.

B. Dial Plan Characteristics

In this section, we characterize the dial plans based on their scale and complexity. Figure 3 depicts the dial plan graph for CO1_NY1 as an example of a typical dial plan. Nodes of the same type are grouped in the same column: CSS, partition, route list, route group, routing device shown from left to right.

As a graph, we measure the dial plan’s scale by looking at its total number of nodes (inclusive of all nodes starting from the directory numbers column to the routing device column) in Table II. For each enterprise, the dial plans are sorted in descending order of number of nodes. The largest dial plan, CO1_IT has more than 10,000 nodes and the smallest dial plan CO1_CA has a little over 100 nodes.

We measure a dial plan’s complexity by its node degree as listed in the last two columns of Table II. The average node degree varies between 1.2 and 2.4, with a max between 8 and 983. Manual configuration and validation of the correctness of all of these nodes and their interconnectivity is a tedious task. The DA VINCI tool described in the next section is designed to help administrators manage their complex dial plans.

IV. THE DA VINCI TOOL

The DA VINCI (Dial plan Analysis, ViSualization aNd Configuration Integrated) tool has two major use cases. First, the designer can use it to extract an existing dial plan from Cisco CallManager, then visualize, simulate and analyze it, so as to improve its design. Second, the designer can use it to create a new dial plan by leveraging the common design patterns discovered from other similar dial plans in the database. DA VINCI supports Cisco CCM/CUCM v. 4-7.

Fig. 4 shows the architecture and workflow of DA VINCI. Its backend functionality include configuration parsing, con-
A summary of configured call routing components in dial plans.

<table>
<thead>
<tr>
<th>Input</th>
<th>Backend</th>
<th>Frontend</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dial Plan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Configuration database, dial plan simplification and simulation, design template generation, dial plan customization and new dial plan generation. Its frontend interface allows the user to visualize, analyze, or configure a dial plan. DA VINCI extracts dial plan configurations from CallManager using its SOAP interfaces (Section IV-A). The tool then parses all configurations and translates them using an XML-based data model. The data in translated format can be used to construct the dial plan graph. All the extracted dial plans are stored in this XML format in an internal database. Through the analysis interface, the user can work with commonly-observed design patterns that are defined as "design templates" (Section V). These templates, combined with other customized designs, allow the designer to quickly generate new dial plans. Once a new dial plan is generated, DA VINCI converts it to native CallManager configurations and configures them in batch mode. The user can also perform analysis to simplify a dial plan design by removing unused configurations and merging similar route/translation patterns (Section VI), or validate its design by simulation. Today, DA VINCI has been used in a number of VoIP deployment projects and has demonstrated its capability in not only significantly simplifying dial plan design and configuration, but also greatly improving the quality and maintainability of the generated dial plan configurations.

A. Dial Plan Extraction

All of the VoIP infrastructures that we have access to are implemented on Cisco VoIP platforms. Thus, the methodology we use to extract dial plans is platform-specific and would need to be adapted to support other platforms.

Cisco CallManager provides a Simple Object Access Protocol (SOAP) interface that we use to extract dial plan configurations. Our extraction works as follows. First, we obtain information about all configured CSSes, partitions, and gateways using the ListCSS, ListPartition, and ListGateway queries. Next, we need to obtain a list of the configured route lists and route groups. However, Cisco does not provide a direct method to access this information. Instead, we take the information obtained from the previous ListPartition and infer the names of the route lists from their associated route/translation patterns. We then use the inferred route list name to query ListRoutePlan, which returns the detailed configuration of the queried component. From each detailed route list configuration, we then identify the associated route groups, and use ListRoutePlan again to retrieve each route group’s detailed configuration. After these steps, we have the information of all configured elements and the associations between them. Then we use the configuration data to create the dial plan graph as described in Section II.

The extraction is a read-only operation that has low impact on the running system. The process can take from a minute to hours to complete depending on the size and complexity of the dial plan. Only users who have the credentials to access CallManager can run the extraction. When users have completed the extraction, they upload the configuration files to the DA VINCI portal for further analysis. Next, we describe the analysis supported in the DA VINCI tool.

V. Common Dial Plan Designs

In order to streamline dial plan design, administrators can reuse common design patterns to implement dial plan changes or new dial plan configurations. In this section, we identify two common design patterns based on two models of utilizing routing resources in multi-site VoIP networks: shared resource routing in Section V-A and local resource routing in Section V-B. Then, we introduce algorithms for identifying these design patterns as subgraphs in the dial plan graph in Section V-C. The DA VINCI tool identifies these patterns in the extracted dial plans and allows users to reuse them for new configurations.

A. Shared Resource Routing Model

An enterprise VoIP network often has multiple voice gateways located at different locations that act as transit points for calls between the VoIP network and the PSTN. Calls destined to the PSTN need to exit the VoIP network at one of these gateways. The choice of gateway for routing a specific call depends on the calling policies implemented in the dial plan. Consider a VoIP cluster that covers two sites, one on the east coast with area code (234) and the other on the west coast with area code (432) in the US, as depicted in Fig. 5(a). To minimize long-distance cost, calls destined to PSTN numbers...
Local resource routing can only be implemented with route patterns or translation patterns. A route pattern implements shared resource routing if and only if it routes calls originating from one site toward a routing device on a different site. For example, in Fig. 6(a), calls originating from partitions 1 to \( n \) but destined to phone numbers with area code (234) (i.e., matches pattern 1234XXXXXXX) are eventually routed to the shared routing device that is not located in any of the originating sites. Note that in this pattern, the first 1 is the country code for the US, 234 represents the area code, and the remaining 7 \( x \)’s can be any 7-digit phone number. Because we are only concerned with the partition site, route pattern, and the routing device, we omit the rest of the dial plan configurations in the figure.

A second implementation of shared resource routing is through the use of translation patterns. A set of translation patterns implement shared resource routing if and only if they are defined for calls originating from different sites but are eventually routed by the same routing device. An example of shared resource routing implemented by translation patterns is shown in Figure 6(b). As the figure shows, calls originating from partitions configured for sites 1 to \( n \) but destined to phone numbers with area code (234) are routed through the same CSS. From this CSS onwards, these calls are routed by the same routing device. Note that the model for shared resource routing based on translation patterns is slightly different from the model based on route patterns as it is more strict with respect to the number pattern. The patterns need to be an exact match. This is because we cannot identify the full call routing path for a translation pattern – we can only identify the path up to its downstream CSS. The rest of the path from the CSS onwards has to be inferred. Without knowing the actual called number, an inference can be made only on patterns that are an exact match.

B. Local Resource Routing Model

Next, we consider another design strategy for call routing where calls from different sites destined to the same phone number exit the IP network at their local gateways. We again consider a VoIP cluster that covers two sites, one on the east coast with area code (234) and the other on the west coast with area code (432), as depicted in Fig. 5(b). Calls from the west coast destined for area code (234) exit to the PSTN at the west coast gateway, instead of exiting at the east coast gateway in the previous design.

Typically, this design is used to support routing for enterprises that do not wish to share local gateways between their different sites, or to support dialing to destination numbers for which cost-savings is not desired such as toll-free numbers. This pattern is also common for calling local services (e.g., 911 emergency call) where only the local gateway with direct access to the PSTN should be used. We term this routing design local resource routing. That is, calls originating from a site are routed using resources local to this site.

Local resource routing can only be implemented with route pattern definitions. The subgraph model for this design is depicted in Fig. 7. In this example, calls that are originated from partitions configured for sites 1 to \( n \) but destined to route pattern 911 are locally routed to each site’s gateway. The routing logic between the route pattern and the routing device may vary for different sites, and are omitted in the figure.

C. Design Pattern Identification via Subgraph Matching

In order to identify the two types of routing designs described above, we develop an algorithm to scan through the dial plan graph defined in Section II. The algorithm works in two steps. First, it extracts, for each translation pattern and route pattern, the associated subgraph that represents the individual routing policy. Then, it compares the subgraph
to the described models to identify those subgraphs that implement the two design patterns. We explain the details of each step as follows.

**Step 1: Extract subgraph for translation/route patterns**

First, the algorithm enumerates all translation and route patterns defined in all partitions. For each pattern defined, it traverses the graph and identifies the subgraph that represents the routing policy implemented. For a translation pattern, the subgraph initially includes the partition that contains this pattern. The algorithm then only traces the dial plan graph one step forward to include the downstream CSS into the subgraph. For a route pattern, the subgraph also starts with the its partition. However, the algorithm traverses the graph multiple steps forward until the subgraph includes the route list, route group(s), routing device(s), i.e., the entire path.

It is worth noting that the subgraph for route patterns needs to contain the routing device intended to route that pattern. However, the subgraph that we have so far contains multiple routing devices that belong to all the route groups in the route list that is configured for this pattern. Because of load balancing, this often results in many routing devices that could possible to be used to route the call. To simplify analysis, we prune the subgraph to only contain links to the primary routing device. The primary routing device is defined in the first route group among the many route groups that may be configured in a route list. In addition, if there are multiple routing devices defined in the first route group, it is the first-choice gateway or trunk in this route group. We prune all non-primary routing devices and their associated route groups from the subgraph.

We use the name and description of partitions (part of the configurations of partitions) to identify distinct enterprise sites. Most of the partitions in the studied dial plans are defined with the location information such as “SiteName Usage Partition”. Similarly, we obtain location information for all routing devices from their names.

After the above processing in this first step, we have the subgraphs that represent the routing policies, defined by translation or route patterns, for all different sites covered by the dial plan, similar to the ones shown in Fig. 6 and 7.

**Step 2: Match subgraphs with design patterns**

To match the subgraphs to the design patterns, we differentiate between the procedure for route patterns and translation patterns. For route patterns the identification is straightforward: for each route pattern, we scan through its subgraph and compare the site of the originating partition and the site of the routing device. If these two sites are the same, the route pattern is classified as implementing shared resource routing; otherwise, it is implementing local resource routing.

For translation patterns, we need to perform an additional grouping step before matching. We group together subgraphs of identical translation patterns together as depicted in Figure 7. With the above grouping, we then perform a search in each group for the subgraphs that ends with the same CSS. The translation patterns defined in these subgraphs are classified as implementing shared resource routing.

<table>
<thead>
<tr>
<th>Dial Plan</th>
<th>Route Pattern Shared</th>
<th>Local</th>
<th>Translation pattern Shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO1JT</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO1JT</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO1JT</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>CO2EU</td>
<td>652/729</td>
<td>37/729</td>
<td>90/0</td>
</tr>
<tr>
<td>TABLE III</td>
<td>NUMBER OF TRANSLATION AND ROUTE PATTERNS THAT IMPLEMENT SHARED RESOURCE ROUTING OR LOCAL RESOURCE ROUTING.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table III summarizes the results from the above subgraph matching. We run the analysis on seven out of the nine dial plans that serve at least two different sites. We find that they all adopt these design patterns, albeit to different extents.

Route patterns shown in Table III are used either for shared resource or local resource routing. As we can see from the table, shared resource routing is extensively used in all dial plans but CO2_AS. In contrast, local resource routing is implemented more often in CO2_AS, accounting for 84 out of 87 route patterns. The other dial plans also use this design, e.g., CO2.EU, CO2_AS, CO3_NA. But in general shared resource routing is more commonly used in defining route patterns than local resource routing.

Translation patterns implementing shared resource routing are not as commonly observed as route patterns. Among the 7 dial plans studied, CO1_JT, CO1_NY2, and CO3_NA did not implement shared resource routing using translation pattern. A special case is the dial plans for CO2, in which more than half of the translation patterns are used to implement this design. For example, CO2.EU has 6 translation patterns that are defined across different sites, 5 of which are observed to implement shared resource routing.

**VI. IMPROVING DIAL PLAN DESIGN**

In this section, we make the configuration more succinct and easier to manage by reducing the number of configuration components while retaining the same call routing logic. Specifically, we identify two common issues in these dial plans: (1) the presence of unused configurations that can be removed in Section VI-A, and (2) the presence of superfluous number patterns that can be merged to simplify the design in Section VI-B. While these mechanisms are meant to improve dial plans, we also need to ensure that they do not change the call routing logic. In order to check that the routing logic remains the same, we use the call routing simulator in the DA VINCI tool.

**A. Unused Configuration Components**

Like configurations in any operational system, dial plans in VoIP system change over time as configuration components are added to support new users, new devices, or implement new routing changes, etc. In some cases, when new configurations are added, old configurations are not cleaned up from the system. In other cases, components may be initially misconfigured.

1In Cisco’s specification, routing devices in a route group are ordered. Different load-balancing algorithms, e.g., round-robin, can be used to choose among these devices for routing.

2Note that the total number of route patterns in this table may be less than those reported in Table II, because here we are only counting patterns that are ‘allowing’ calls, not those ‘blocking’ patterns.
and have never been used. These unused configurations make a dial plan unnecessarily complex. Furthermore, it could make future dial plan modifications error-prone.

Finding unused components in a sea of thousands of configuration components in any given dial plan is non-trivial. We define an unused configuration as one in which a dial plan component (CSS, partition, etc.) is either not fully specified, or not properly configured to work with other components to implement usable call routing logic. Using the dial plan graph developed earlier, we approach the problem as follows.

We observe that up to 65% CSSes (for CO3_US) and 50% partitions (CO2_AS) were not used. Other components, such as route lists, route groups, or routing devices, are less error-prone, because they are directly mapped to physical resources (e.g., gateways, trunks, etc.). For example, if a gateway is not properly configured, it will not be used in the system. Such problems are more likely to be identified and corrected in the initial deployment rather than in an operational system. Nevertheless, we were able to find several unused configurations for these components.

In our study, we found that some of these dial plans have been in use for over five years and have significantly evolved over time. Without the model and analysis we develop in this paper, it would be a tedious manual undertaking to identify any of these unused configurations.

### B. Superfluous Number Patterns

In this section, we look at improving the design by creating simpler dial plans with fewer configuration components. While the definition of a good or bad routing design may be subjective, a generally useful principle for simplification is to produce a smaller dial plan out of an existing dial plan while still implementing the same set of routing policies. In this section, we study the problem of identifying superfluous number patterns. Specifically, we search for instances of route patterns and translation patterns that implement the same call routing policy, and can be merged to reduce the total number of configuration components.

We note that the set of potential route or translation patterns that can be merged needs to satisfy several conditions. First, they must belong to the same partition. Second, route patterns must point to the same route list, or routing device, and translation patterns must point to the same CSS. Third, they must process the call in the same manner. That is, calls matching the mergeable patterns must be either all allowed or all blocked, and the calling/called digits are transformed in exactly the same way. We call patterns that satisfy these three conditions similar patterns.

Similar patterns are not always necessarily mergeable. The additional condition for merging is that these set of similar patterns must be defined as a single pattern using Cisco’s regular expression syntax [7]. For example, patterns 1xxx and 22xx are not mergeable, while 12xx and 22xx are mergeable as [1, 2]2xx. We define that two patterns are mergeable, if and only if they are similar patterns that have the same digit length and only differ from each other by one digit. Based on this definition, we develop a simple algorithm using heuristics.

<table>
<thead>
<tr>
<th>CSS</th>
<th>In-Degree</th>
<th>Out-Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO1</td>
<td>10 (36%)</td>
<td>6 (22%)</td>
</tr>
<tr>
<td>CO2</td>
<td>7 (27%)</td>
<td>4 (15%)</td>
</tr>
<tr>
<td>CO3</td>
<td>5 (18%)</td>
<td>3 (10%)</td>
</tr>
<tr>
<td>CO4</td>
<td>3 (11%)</td>
<td>2 (7%)</td>
</tr>
<tr>
<td>CO5</td>
<td>2 (7%)</td>
<td>1 (3%)</td>
</tr>
<tr>
<td>CO6</td>
<td>1 (3%)</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>

**TABLE IV**
Degree Specification for Nodes to be Considered Used for Call Routing. For a Routing Device to be Considered Used, It Must Have at Least One In-Degree or One Out-Degree Configured.

Intuitively, each node that represents a dial plan component in the graph needs to be on at least one complete call routing path. Figure 2 illustrates some of the call routing paths that can possibly be specified in a dial plan. In order to be part of a path, each must meet certain requirements on its minimum or maximum in-degrees and out-degrees. We specify that minimum and maximum degrees in Table IV. For most nodes, such as a CSS, or Route List, etc., they must have at least one incoming edge and at least one outgoing edge. A Partition must have at least one incoming edge from a CSS and at least one outgoing edge to Translation Pattern, Route Pattern or Directory Number, and so on. In this table, we use $[a, b]$ to denote a minimum degree requirement of $a$ and maximum degree requirement of $b$, and $[a, \infty)$ to denote the case where a node only has a minimum degree requirement.

We note that the condition for a routing device is special. A routing device can be configured to be an incoming-only device or out-going only device. Therefore, it must have at least one incoming or one outgoing edge but not necessarily both.

Using this notation, we analyze the dial plan graphs and identify the unused nodes. We first identify the nodes that do not meet the in-degree or out-degree requirements. Then we remove these nodes and all related edges from the graph, and re-examine the residual graph for other unused nodes. This process continues iteratively, until the graph can no longer be pruned.

### A summary of all unused configuration components for the dial plans are listed in Table V. Note that a partition is considered unused if at least one of its route patterns, translation patterns, or directory numbers are not part of a complete path. From the results, we see that in all the studied dial plans, CSSes and Partitions have the most unused components. This is not surprising because the partitions are the most complex to configure in a dial plan. Unused CSSes are also common because they are often configured for testing purposes and not removed even if they are no longer in use.

**TABLE V**
Number of Detected Unused Configuration Components.
to merge patterns. The algorithm first scans through the dial plan to identify all sets of similar patterns. Then, for each set of similar patterns, it iteratively merges pairs of patterns with the same length, until no more pairs of patterns are mergeable.

We apply our merging heuristic to the extracted dial plans and find that superfluous route or translation patterns commonly exist. Figure 8 depicts the number of route patterns and translation patterns, before and after the merge for each partition, in the CO1-NY1 dial plan. Overall, we achieved significant simplification and reduction in number patterns – 36% route patterns and 9% translation patterns are reduced after merging. As part of future work, we are improving this heuristic.

VII. RELATED WORK

There has been a lot of work on measuring the traffic and quantifying the performance of VoIP networks and systems [8], [9], [10]. However, there has been little attention paid to the configuration of call routing in VoIP networks. Closer to our work, [11] discusses some aspects of dial plan configurations in PSTN vs. VoIP networks, while [12] uses Call Detail Record (CDR) data to infer the call routing policies implemented in a VoIP network. Our work differs from the previous work in that we focus on analyzing dial plans from operational networks, identifying common design patterns, and improving the design while maintaining the correctness of the implemented routing policies.

In the area of systems and network configuration management, [13] formalized and automated model finding in network configurations, which is similar to our effort in identifying common design patterns. [14] proposed the concept of “shadow configuration”, which applies the same “simulate-then-configure” principle adopted by our DA VINCI tool. Also, automated configuration has also been extensively studied [15], [16], [17]. While our work shares the same motivation as these related work, we address the problem in a different domain with distinct issues and challenges.

VIII. SUMMARY

In this paper, we developed the DA VINCI tool to help improve the efficiency of the VoIP management through extraction, analysis, simulation, and stream-lined configuration of call routing. We analyzed the call routing configurations of nine operational enterprise VoIP networks to understand the complexity of call routing design and enabled administrators to better understand and reuse existing designs. We modeled the dial plan design as a directed graph and conducted analysis to reveal the common design patterns embedded in the graphs, i.e., shared resource routing and local resource routing. We found that all dial plans adopted both design patterns. However, shared resource routing is the more popular design, as it typically provides cost savings to enterprises. Lastly, we looked at a couple of approaches to identify and eliminate unused or superfluous configurations in the dial plan, as it evolves over time, to improve the quality of its design. We found many calling search spaces and partitions that were configured in the system but not used for call routing, and therefore could be removed from the system. We also found many route patterns and translation patterns that can be merged to simplify the dial plan design.

REFERENCES