

Revenue Recovery for Operational SONET/SDH Networks

Wee Teck Ng, Pankaj Risbood and Swarup Acharya

Bell Laboratories, Lucent Technologies, Inc., 600 Mountain Avenue, Murray Hill, New Jersey 07922, USA
{weeteck,riskoob,acharya}@lucent.com

Edward Lafontaine

Lucent Worldwide Services, Lucent Technologies, Inc., 121-052 1 Robbins Road, Westford, MA 01886, USA
elafontaine@lucent.com

Abstract: Large SONET/SDH networks typically have 20-30% “stranded” resources, both in wasted bandwidth and in untracked hardware. We present a novel approach that uses localized heuristics to uncover stranded resources and configuration errors based on *field data alone*. Unlike traditional solution which requires complex reconciliations between field data and various inventory databases, our approach synthesizes higher-level models from field data and analyzes the models to identify stranded resources and configuration errors. Field trial results indicate that our approach is very effective. For example, we found that 33% of 1+1 protection circuits have incorrect configurations impacting service reliability, and 10% of the circuits in a large carrier network have stranded bandwidth that can be “recovered”.

©2005 Optical Society of America

OCIS codes: (999.9999) SONET/SDH network management; (999.9999) inventory reconciliation

1. Introduction

SONET (Synchronous Optical Network) and SDH (Synchronous Digital Hierarchy) networks are widely deployed in enterprise and telecom networks as access and backbone networks [1]. But the inventory systems for SONET/SDH networks are typically only 60-80% accurate [2]. This results in many erroneous configurations that consume network resources without serving customers or generating revenue. Fig. 1 shows an example of a stranded circuit fragment that was not deleted when the service was deactivated. Since this circuit fragment does not exist in the carrier’s inventory system, an operator will try to use this resource to setup new services but all such attempts will fail. Stranded resources can also result when undocumented network elements and cards are not reflected in the inventory system. The network operator may not be aware of the availability of these resources to support new services, resulting in the unnecessary deployment of additional equipment. RHK estimates that the typical service provider has 20-30% stranded assets at any given time [3]. Stranded resources can also lead to longer service provisioning times because a network operator cannot determine the status and capacity of network resources in a timely fashion [4].

In our studies, we have found that carrier networks also have a significant amount of incorrectly configured equipment that affects network reliability and performance and not often the focus of current approaches [4][5]. These configuration errors cause latent errors and are hard to detect because they are dormant until another error triggers them. This may result in degraded reliability than that promised in the service level agreement and can potentially impact a carrier’s revenue.

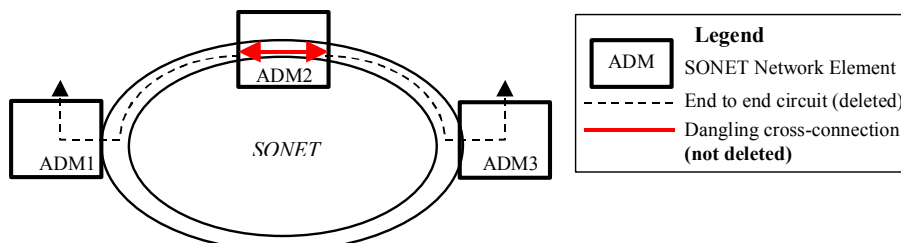


Fig. 1. Dangling cross-connect resulting in stranded resources. A circuit was originally provisioned from ADM1 to ADM3 and not properly deleted, leaving a circuit fragment in ADM2.

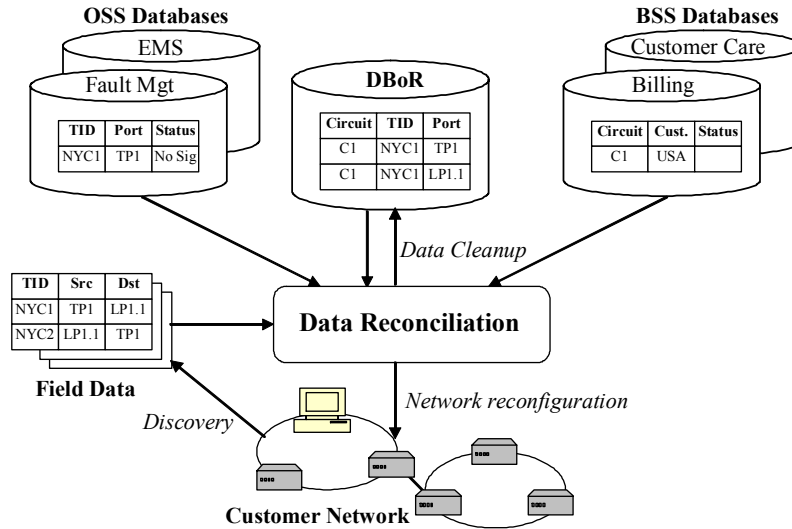


Fig. 2. Identifying stranded resources via inventory reconciliation.

Stranded resources and configuration errors are caused by operator errors, prevalence of dumb devices with no management interfaces (e.g. patch panels), software bugs, and management process deficiencies and cannot be easily prevented [5]. The traditional solution to identify stranded resources is to improve the accuracy of the inventory database using an inventory reconciliation system, and subsequently check the updated inventory database for errors and stranded resources [4][5]. Fig. 2 shows a typical inventory reconciliation process. One first gathers up-to-date data from the SONET/SDH equipment (i.e. field data) and normalizes it to some format (e.g. relational database). The field data is then compared with normalized data extracted from various databases in the data reconciliation phase. This is a complex and expensive process as it works on multiple databases with different access methods and schema. A typical telecom carrier has many management systems, including a database of record (DBoR) which stores the inventory data, Business Support Systems (BSS) like billing and customer care, and Operation Support Systems (OSS) that manages the SONET/SDH equipment like Element Management System (EMS).

Many telecom carriers are experiencing budget constraints and are not willing to invest heavily in an inventory reconciliation system, especially on legacy SONET networks that may be replaced in a few years. These customers are looking for a low-cost approach to quickly identify stranded resources. In this paper, we propose a simpler and more direct approach to find a large class of stranded resources and configuration errors from *field data alone*. These errors include unused resources, excessive sparing, incorrect (but legal) configurations and inefficient configurations. Our approach is not meant to be as exhaustive as the standard reconciliation approach; instead for a provider not inclined to invest heavily, it aims to provide a low-cost approach to remove some of the most common and glaring inefficiencies in the system.

Our approach is depicted in Fig. 3. Like a traditional reconciliation system, we also need to gather field data directly from SONET/SDH equipment. We model the network in extensible markup language (XML) [6] and perform various types of data analysis and synthesis to identify stranded resources and configuration errors. Our approach is novel in the heuristics we developed to detect stranded resources and configuration errors. We use our understanding of the network to develop a set of semantics-based validators to detect incorrect configurations and stranded resources. We also develop a status-based approach to detect stranded resources by comparing the synthesized data with equipment status and alarm data. Our approach is significantly faster than conventional data reconciliation tools and can be run frequently to keep the network “clean”. In many cases, our approach is the only feasible option as only the field data is available. Moreover, our approach can complement existing data reconciliation tools as it can be used as a front-end tool to greatly reduce the volume of corrupt data into the reconciliation tool. Field trial results indicate that our approach can identify a significant amount of stranded resources and configuration problems. For example, we found that 10% of the circuits in a large customer network are stranded circuits, and 33% of 1+1 protection circuits have incorrect configurations.

1.1 Contributions

The contributions of this paper are as follows:

1. This paper highlights the problem of stranded capacity in service provider networks and the potential monetary and efficiency benefits in recovering this capacity.
2. We list specific provisioning and equipment problems one might encounter in a typical service provider SONET/SDH transport network.
3. We propose an effective, low-cost approach based on the network field data with a goal to correct any specific, localized network problems where they are known to exist. Our approach encompasses both syntactic and semantic analysis based on creating high-level circuit and network models from the underlying data.
4. We present summaries of field trials of our approach and highlight the bandwidth (and consequent, revenue) recovery that has been achieved.

1.2 Outline

The rest of the paper is organized as follows. We give an overview of our approach in Section 1.2.1. In Section 2, we describe our data discovery process and data model. In Section 3, we list the various stranded resources and configuration errors we address and describe how we identify them. In Section 4, we show how to synthesize higher-level models from base constructs and use them to identify more stranded resources. In Section 5, we present results from two studies on operational SONET network. In Section 6 we discuss the limitations of our approach and identify networks where our approach works best. In Section 7, we discuss related work and finally, we conclude in Section 8.

1.2.1 Process Overview

We begin our study by collecting detailed network information directly from the SONET/SDH network element (NE) and modeling the network using a XML language (Step 1 in Fig. 3). Next, we examine the field data to identify syntax and semantic errors on base constructs (Step 2). The base constructs are the ports, timeslots and cross-connections that make up a SONET/SDH network. We then synthesize higher-level models (i.e. circuits, rings and NE) from the data and analyze this synthesized data to identify stranded resources and configuration errors (Step 3). The network is reconfigured to remove these errors (Step 4).

In this paper, we use BLSR (Bi-directional Line-Switched Ring) OC-48 SONET [1] rings as examples but our approach is applicable to SONET and SDH networks of different multiplexing hierarchies and configurations (e.g. ring, mesh). We focus on the *cross-connection* base construct and the *circuit* model to illustrate our algorithms. Other base constructs and higher-level models have similar or lower complexity.

2. Data Collection and Network Modelling

Field data can be collected manually by an operator or automatically using a network discovery system [5][7]. We describe the former approach here. It consists of retrieving the network topology, circuit pack inventory, equipment configuration and operational status via the SONET NE's management interface. The

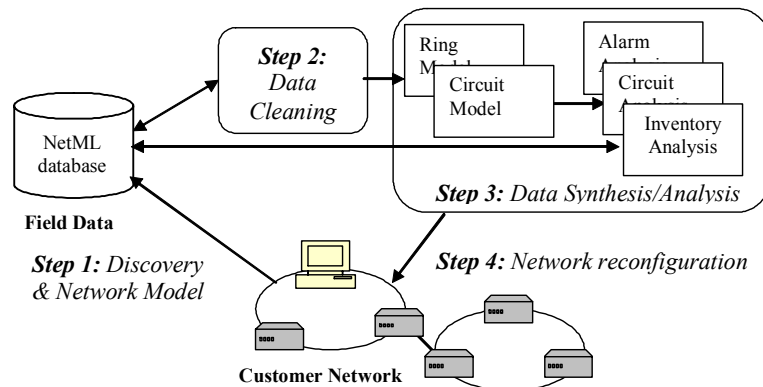


Fig. 3. Identifying stranded resources via data synthesis and analysis.

field data can be in various formats like Transactional Language 1 (TL1) [8] and Craft Interface Terminal (CIT) [9]. We convert the native format to a normalized format and store it in a database for further analysis.

We model the network using NetML [6]. NetML is a XML-based language for modeling computer and telecommunication networks. Unlike most network management system that model different level or aspect of the network and store the data in different databases (relational, XML, IMS, etc), we use NetML as a common central database to model all aspects of the network. NetML also has a rich set of analysis and visualization tools to expedite network analysis and research, and we can also leverage many powerful public domain packages like XSLT and XQuery languages.

Fig. 4 shows an example of the raw data format and NetML model of an OC48 NE. We first briefly describe the functionality of a SONET/SDH NE and some industry terminology we use (for a deeper exploration, the reader is referred to [1]). The role of a SONET/SDH NE is to be a data multiplexor, collating numerous low bandwidth (or, “low-speed”) streams (e.g., STS-1/52Mbps) of customer traffic into a single high bandwidth, or, “high-speed” stream (e.g., OC-48/2.5Gbps) that is carried long distances. Consequently, an NE consists of numerous “client-side” ports or, *low-speed* ports and one (or, at most very few) “line-side”, or, *high-speed* ports. The multiplexing hierarchy is different for SONET and SDH --- the former has STS/OC-1,3,12,48 bandwidth rates while the latter follows the STM-1,4,16 hierarchy and Fig. 4 shows it for the SONET case. Each NE typically is part of the “ring” with a minimum of two pairs of fibers --- one carrying traffic in the clockwise direction and the other in the counter-clockwise direction. The neighboring nodes are often referred to as being in the *East* and *West* direction.

The data files in Fig. 4 show a STS1 cross-connection between a low-speed port (5a-1) and a STS1 tributary (1w-1-1) on the OC48 high-speed link. The raw data is in TL1 format. The pictorial representation is generated using NetML’s *nplot* and *BLLinkBrowser* tools [6].

3. Data Cleaning

Once we have a complete model of the network we examine the network model to “clean” erroneous data. Data cleaning is traditionally used to remove “dirty” data resulting from incorrect abbreviations (e.g. Stby instead of Stdby), data entry mistakes (e.g. NY instead of NYC), duplication, missing fields, etc [11][12]. In our study we go beyond standard syntax and schema checks to detect semantic errors too.

Data cleaning is done in a pipeline approach. We begin by checking for syntax errors and model/schema constraint violations in the base-constructs. Next, we look at the semantics of the data and check for configuration errors. We then use status information to look for more configuration errors and unused resources.

3.1 Syntax check

We perform a syntax check on the raw data before converting it to NetML. These checks are data-format specific and are part of the data parsing process. For example we verify that a DS3 low-speed port has the proper syntax (i.e. ls-[1-8][a|b]-[1-3]) [9].

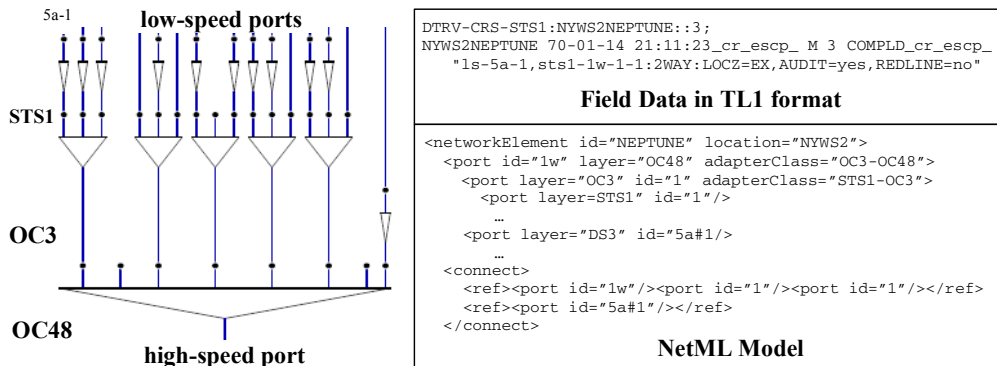


Fig. 4. Raw data and NetML representation of an OC48 NE named NEPTUNE.

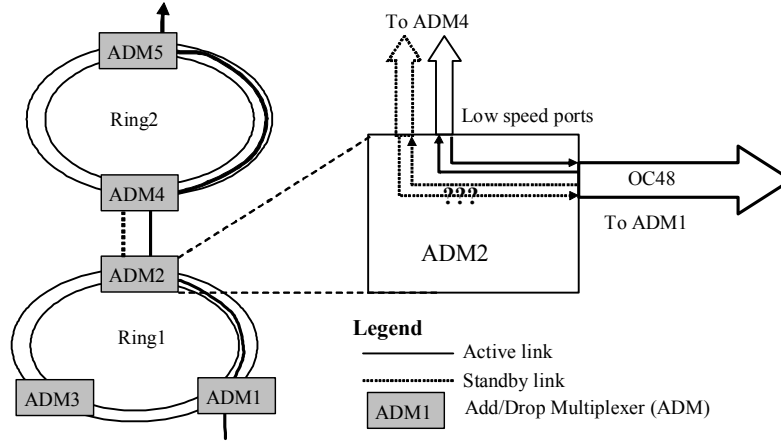


Fig. 5. Invalid 1+1 cross-connection. ADM2 has a missing cross-connection marked ??? thus causing the standby link to have only 1-way communications when the active link fails.

3.2 Data model checks

After the raw data is converted to NetML, we check the NetML model for schema and semantic errors. The latter validates the semantic properties of a NetML file that cannot be easily checked by XML schema validation. The basic algorithm to clean cross-connection is as follow:

Definition 1. A cross-connection X is defined as a tuple $(EP(X), WP(X), EA(X), WA(X), ES(X), WS(X), type, timeslot)$. Here, E and W refer to East and West and P , A and X refer to Port, Address and Status respectively.

1. EP and WP represent one of two things. They are either low speed ports (e.g. 5a-1 in Fig. 4), or, tributaries on the high-speed port (e.g. 1w-1-1 in Fig. 4) and are ordered east to west. For example, (5a-1, 1w-1-1) in Fig. 4.
2. EA and WA are equipment names (e.g. NYW2NEPTUNE in Fig. 4) and are ordered east to west.
3. ES and WS are port status and are ordered east to west.
4. $type$ is the cross-connection type (e.g. one way, two way, active, standby).
5. $timeslot$ is the timeslot on the high-speed port (e.g. 1-1 in Fig. 4).

NetML will verify that EP and WP exist and are at the same multiplexing hierarchy, and $EP(X) \neq WP(X)$, $EA(X) \neq WA(X)$ (i.e. no loopback). For example, it will flag an error when it finds a STS1 cross-connection between a DS3 port and a STS3 link.

3.3 Semantics-based cleaning

We use our understanding of the semantics of the data to verify that the basic constructs are valid. For cross-connections, we check the type, address and configuration rules. Type field validators are used to detect latent errors that are caused by legal but inconsistent cross-connection configurations, such as the 1+1 protected cross-connection validator shown in Fig. 5. Fig. 5 shows a circuit originating at the SONET Add/Drop Multiplexer (ADM) named ADM1 in Ring1 and terminating at ADM5 in Ring2. This circuit requires 4 1-way cross-connections in ADM2 and ADM4 to support an active and a standby two-way links. The circuit will operate even if the standby link is not provisioned or partially provisioned as shown in Fig. 5, but will fail when the active link between ADM2 and ADM4 fails. This type of latent error has severe reliability implications on the customer as it downgrades the circuit's reliability from 99.9999% to 99.9%¹. Since these errors do not manifest themselves at the time of provisioning a service they are quite common in operational networks. We found that 33% of the 1+1 cross-connections in a large customer network have this type of errors.

¹ Assuming links fail independently, a single link will fail with probability p , and a 1+1 link will fail with probability p^2 . Availability is defined as $(1-p)*100\%$, so if $p=0.001$ (i.e. 1 link failure a year with mean repair time of 8 hours), the availability of a single and 1+1 links are 99.9% and 99.9999%, respectively.

```

Input: A list of cross-connections  $X_i$  of a network element,  $i=1..n$ 
Output: A list of invalid cross-connection
for  $i=1$  to  $n$  do List[ $i$ ] = [] // initialize empty list
for  $i=1$  to  $n$  do List[EP( $X_i$ )].append( $X_i$ ) // create cross-connection list
for port in List do
   $X_o$  = List.pop()
  if List.length() == 1 // A single 1-way crossconnection
    if type( $X$ ) == oneway
      Print "oneway crossconnection ", X.print()
  else
    for  $i=2$  to List.length() do
       $X$  = List.pop()
      if type( $X$ ) != type( $X_o$ ) // inconsistent crossconnection
        Print "inconsistent crossconnections ", X.print()

```

Fig. 6. Algorithm to detect invalid cross-connection types originating from low-speed port.

The basic algorithm to detect inconsistent cross-connection types is shown in Fig. 6. We group the cross-connection into lists indexed by the low-speed ports, and inspect the list for inconsistent cross-connection types (e.g. a 1+1 cross-connection has a mixture of 1-way and 2-way cross connections). In practice we use the XQuery language to implement the validators for convenience and maintainability [10].

3.4 Status-based cleaning

An extension of the semantics-based cleaning is to use status information to identify stranded resources and configuration errors. We can examine the status of each base construct to verify that it is carrying live traffic. For example, we can flag a cross-connection as unused if it does not carry any user traffic. Conversely, a low-speed port that is carrying user traffic but has no cross-connection configured is missing a cross-connection. Sometimes the status information may be vague (e.g. a port may be configured but not set to monitor errors) and we will need more information to deduce if resource is in use. In Section 4.1.2, we will provide more detail on status-based cleaning and circuit analysis.

The next section describes how we synthesize a high-level model from base constructs to filter the error data and to deduce more conclusively if a base construct is in use or incorrectly configured.

4. Data Synthesis/Analysis

The data cleaning performed in Section 3 is local to NE. In this section, we synthesize higher-level constructs from base-constructs to do data cleaning and analysis on a ring-wide basis. This allows us to uncover more stranded resources and configuration errors than that found in Section 3. The higher-level constructs are circuits, rings and NE model. We use these constructs to perform circuit and inventory analysis.

4.1 Circuit analysis

We define a circuit as an end-to-end path between two low-speed ports in a single SONET ring:

Definition 2. A circuit C is defined as a list of cross-connections (X_1, X_2, \dots, X_n) such that $timeslot(X_i) = timeslot(X_j)$, $i, j = 1..n$ and $X_i \in Ring$.

Fig. 7 shows 3 STS1 circuits between ADM1 and ADM2. Both are part of an OC48 BLSR ring. In this

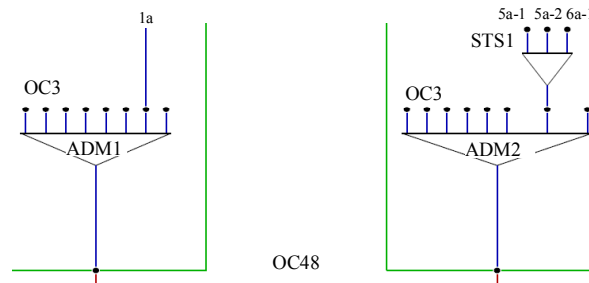


Fig. 7. Circuit definition example.

example the OC48 link between ADM1 and ADM2 is de-multiplexed into 8 lower speed OC3 links. An OC3 link on timeslot 7 of the OC48 link is further de-multiplexed into 3xSTS1 circuits between low-speed ports 1a-1, 1a-2, 1a-3 of ADM1, and low-speed ports 5a-1, 5a-2, and 6a-1 of ADM2, respectively. Since a typical customer circuit spans multiple rings and may encounter a dumb device like a patch-panel when it jumps from one ring to other, for our analysis we define circuit as a fragment of end-to-end customer circuit contained in a single ring. Once the circuits are extracted from field data we can perform semantics and status-based cleaning.

4.1.1 Semantics-based circuit analysis

In semantics-based cleaning we are verifying that a circuit is correctly configured. A number of different errors can be caught during this process for the circuits in the data. Due to space reasons, we present four classes of errors.

Dangling Circuits: An example of a dangling circuit was presented in Section 1 (see Fig. 1). Essentially, a circuit is dangling if either end of the circuit does not terminate on a low-speed port, i.e., one of $EP(X_1)$ and $EP(X_n)$ is not a low-speed port, where X_1 and X_n are the end cross connects.

Squelch circuit: A squelch circuit has incorrect addresses on one or more cross-connections along its path. It is an example of a latent error as the addresses are only used during SONET ring recovery, but has severe implications on the customer's SLA during ring/node failures.

Trunk Circuits: A correct circuit may be configured in a non-optimal way by consuming unnecessary resources or using non-optimal timeslot layout on intermediate hops. The former is very common on networks with heterogeneous equipment and is detected by the trunk circuit validator. For example a STS1 circuit may traverse multiple ADMs with different switching granularities, and will consume a larger STS3 bandwidth on ADMs that cannot switch at STS1 granularity. These circuits could potentially be re-routed to ADMs with matching switching granularity.

Link and Ring Defragmentation: A SONET ring can also have non-optimal timeslot layout when circuits are added and deleted over time. This causes the available bandwidth on the ring to become fragmented leaving behind "stranded" capacity which may be difficult to use while provisioning circuits. Fragmentation occurs at all levels of the network – in links, rings and the mesh network. The fragmentation validator will detect link fragmentation. For example, a SONET ring may have sufficient capacity to support an OC12 circuit, but cannot provision an OC12 circuit if these twelve slots are in non-contiguous locations. We have developed algorithms to address these optimality issues in [13][14].

4.1.2 Status-based circuit analysis

After removing invalid circuits based on semantics, we examine the port status of a circuit's end-points. The most common port statuses are in-use (i), not in-use (a) and not monitored (n). The last status means the network operator has disabled the diagnostic capabilities of a low-speed port. In many operational networks this indicates an unused port as technicians frequently turn-off diagnostic capabilities of unused port to suppress error messages. In our study we use the following rules to classify circuit status. Given a circuit C with cross-connections X_i ordered from east to west, we examine the low-speed port status of the circuit end points (X_1 and X_n) as follow:

1. **Liveness.** A live circuit has one or both end points in-use, $ES(X_1) = WS(X_n) = i$.
2. **Unused.** An unused circuit has both end points not in-use, $ES(X_1) = WS(X_n) = a$.
3. **Unknown.** Otherwise we mark the circuit as possibly unused and require further investigation. Ports that are not monitored must have its monitoring capabilities enabled to allow further analysis.

4.2 Inventory analysis

The SONET/SDH field data contains detailed configuration information that can be used for inventory analysis. This includes examining each ADM to see if it has adequate or excessive sparing, and verifying its software configuration.

We construct a logical model of an ADM, which has n low-speed ports, m high-speed ports (see Fig. 4) and k spare ports. The high-speed ports are used to construct a ring network, and the low-speed ports contain circuit packs that multiplex incoming traffic to the high-speed port. Important traffic like

emergency 911 voice circuits are typically protected via a $1 \times n$ circuit pack in the spare port. We can deduce if an ADM has excessive or inadequate sparing from the logical model by examining the circuit pack configurations in the low-speed and spare ports.

Improper software configuration can potentially disrupt network operation and cause intermittent faults. We also verify the software configuration of each ADM to ensure that it has the proper software configuration (e.g. OS release, patch level, time) and the configuration is consistent across a ring.

5. Sample Result and Field Studies

In this section, we present results from studies on two operational SONET networks. We found significant amount of stranded resources and configuration errors in both networks. We also identify many areas for optimization, and present the results from a revenue recovery study in Section 5.3.

5.1 Network A

The first study was conducted on a large carrier network in USA. We were allowed to gather field data directly from the SONET NEs for one regional network. The network consisted of 57 OC48 SONET BLSR rings spanning 3 states, and an average ring size of 3.1 ADMs. There are 1372 circuits on the OC48 rings, consuming 69% of the available bandwidth. The ring utilization was bimodal, with many rings heavily utilized and some rings lightly utilized. There are also 30 subtending OC12 access rings supporting 141 circuits.

Prior to our study the operator reported significant provisioning problems. One out of three circuit provisioning attempts failed even though the DBoR indicated that the network has sufficient capacity. When we analyzed the field data we found significant stranded resources in the network:

- 2.1% dangling circuits, consuming 2.4% of the total bandwidth.
- 3.8% provisioned but unused circuits, consuming 3.2% of the total bandwidth.
- 4.4% provisioned and probably unused (not monitored) circuits, consuming 4.8% of the total bandwidth.
- 14.5% of ADMs had excess circuit packs that could be removed and deployed elsewhere.

We also found many configuration errors affecting network reliability and performance, including:

- 32% of circuits expected to have 1+1 cross-connections were incorrectly provisioned. This corresponds to 1% of all circuits.
- 12% of ADMs have older OS or incorrect software configuration.
- 13% of ADMs have insufficient circuit pack protection. Some of these ADMs were known to be carrying traffic for emergency services.

5.2 Network B

The second study was conducted on a metropolitan network in a large city. The network consists of 103 OC48 SONET BLSR rings and an average ring size of 3 ADMs. There are 2155 circuits consuming 75% of the available bandwidth. In this study we were given partial field data that excludes equipment status and configuration information because of which we were unable to perform status, inventory and many types of semantics-based analysis. Our data cleaning was done off-line.

Given these restrictions we still found some amount of stranded resources and configuration errors. In particular, we found 1% dangling circuits consuming 1.2% of the total bandwidth. We also detected 1.4% squelch circuits and 0.2% circuits with invalid addresses. Both are latent errors that could potentially impact failure recovery.

5.3 Revenue Recovery Study

We now present the result of the revenue recovery study on a sample ring (it is one of the rings from the study in Section 5.1). Fig. 8 shows the ring layout pictorially. It is a 6-node OC-48 BLSR ring with the x-axis denoting the nodes and the SONET slots on the y-axis. Only the first 24 slots (of the 48 in the ring) are shown since in BLSR, half the ring is used for protection. A colored tab indicates that the particular slot is carrying traffic between the two nodes, or, it is empty otherwise. There were 44 circuits of STS-1 granularity on the ring.

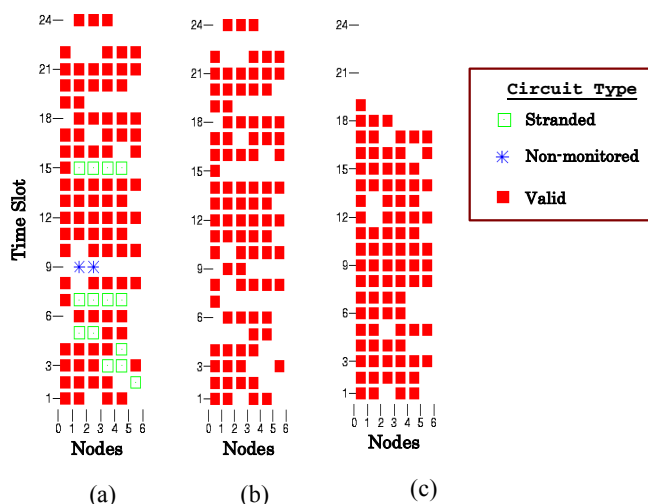


Fig. 8. Sample output of cleanup study. (a) Original ring with stranded and non-monitored circuits (b) Ring after stranded circuits are cleaned (c) Ring after re-routing to defragment space.

Recall that the first step of the process is to do a “cleanup” of the data that eliminates obvious syntactic errors (e.g., Stby instead of Stdb). Then, we perform a circuit analysis to classify the nature of all circuits in the ring and the colors indicate the classification of cross-connects after our data analysis, namely, valid, stranded, or, non-monitored (Fig. 8(a)). Fig. 8 (b) shows the output of the cleanup process described in the paper. In this case, all the stranded cross-connects were removed which accounted for 9.7% bandwidth for this specific ring. For a service provider, recovering 10% of the bandwidth is a huge operational and monetary savings. For this specific case, it was determined that the non-monitored circuits were indeed valid circuits and thus, not deleted from the network.

For both the two field trials, estimates of monetary savings from improved provisioning time and additional “freed” capacity were also recorded. For example, 2.4% of additional capacity available in a regional network for a large service provider translates to hundreds of thousands of dollars per year. Moreover, in many cases where the service provider plans to upgrade the network to next-generation elements, an accurate inventory of the actual traffic lowers their migration cost and enables them to reduce the size of the new network ordered.

Further savings could had by defragmenting the bandwidth on the ring and this is shown in Fig. 8 (c). Bandwidth defragmentation was briefly mentioned in Section 4.1.1 and the reader is referred to [13][14] for a complete description of this process. This process aims to coalesce all the stranded capacity into contiguous locations. The figure shows the layout of the ring after circuits have been re-routed one-at-a-time so as to create large contiguous free spaces. For example, a new STS-3 demand, requiring three contiguous timeslots, from node 2 to 4 can now be met which would not have been possible otherwise. This new layout was achieved by re-routing 12 circuits (out of the 44 in the ring) to new routes. While the focus of this paper is on the data cleanup process, this example summarizes of the end-to-end revenue recovery process.

6. Limitations of Our Approach

Although our approach is significantly cheaper and faster than traditional tools, we are not as exhaustive as existing tools. For example, we cannot detect the following three types of circuits that would be of interest to the service provider: circuits that are lacking status information, live circuits owned by delinquent customers, and certain type of valid circuits that are configured incorrectly (e.g. a customer pays for a 1-way video broadcast circuit but a 2-way circuit is configured instead). The first type of error can be overcome by requiring the carrier to enable port monitoring prior to the study. This is also a good engineering practice [9].

Our approach is most useful in networks where the DBoR is correct and the field data is wrong. These networks are characterized by many failed provisioning attempts (e.g. network A in Section 5.1), and cleaning the field data will remove the stranded resources as well as improve the accuracy of the DBoR.

We found from field trial results, customer discussions and published reports that these networks are the most common.

Our approach is partially applicable in cases where both the inventory system and the field data are wrong. In this case, our approach can clean the field data and simplify future data reconciliation attempts. Our approach is not useful if the DBoR is wrong and the field data is correct. This is not a likely occurrence since the carrier would have discovered the issue from customer complaints.

7. Related Work

Existing data reconciliation tools from vendors like CoManage (www.comanage.net), MetaSolv (www.metasolv.com) and Granite (www.granite.com) typically follow the model in Figure 2. The main goal is to get the DBoR up-to-date and perform stranded resource recovery from the DBoR. Some tools also perform data synthesis or assimilation [4] to reduce the amount of data that needs to be reconciled. In contrast, we skip the data reconciliation step and operate on the field data directly. Our approach is faster and more effective for the investment made as it operates on critical portions of the network where problems are already identified. Moreover, we can use heuristics that can account for specific, localized engineering rules.

Traditional reconciliation tools can improve the inventory system accuracy to 80-95%, but cannot eliminate all errors [4][5]. This is because the service provider network typically has many dumb devices like patch panels that are managed manually and are thus prone to human errors.

8. Conclusion and Future Work

In this study we proposed a simple approach to identify stranded resources and configuration errors in SONET/SDH network. Our approach is based on data synthesis and analysis of field data only and is widely applicable to most SONET/SDH networks. We present field trial results that show that our approach is simpler and less costly than traditional data reconciliation tools. We are able to identify a large class of stranded resources and configuration errors on operational networks. Our approach also complements existing data reconciliation tools as we can greatly reduce the number of inconsistencies between the field data and DBoR.

We are also exploring how we can extend our approach to other types of legacy networks (e.g. ATM and telephony networks). These networks have significant inventory accuracy problems [2][5] and can benefit from our low-cost approach.

9. Acknowledgements

We would like to thank Philip Bohannon for his helpful feedback on earlier versions of this paper, Steve Fortune for supplying the NetML tools used in this study, and Lawrence Cowsar for contributing to the ideas presented in this paper.

10. Reference

- [1] W. Goralski, "SONET", McGraw-Hill Osborne Media, Emeryville, CA, USA, 17 May 2000.
- [2] S. Mewada, "Data Integrity Management: Managing Business-Critical Data Across CSPs' Stovepipes", Yankee Group Report, Jan 2005.
- [3] *MetaSolv Inventory and Reconciliation*, (MetaSolv, 2005), <http://www.metasolv.com>
- [4] *An Overview of TrueSource*, (CoManage, 2003), <http://www.comanage.com>
- [5] L. Goldman, "Discovery and Reconciliation Portend Dramatic Improvements in Telecom Operational Efficiency", RHK Report, June 2003.
- [6] P. Bohannon, S. Fortune, C. Martin, "The NetML Network Model", Lucent Technologies Technical Document ITD-04-45374V, 26 May 2004.
- [7] Y. Breitbart, et al. "Topology Discovery in Heterogeneous IP Networks", in *Proc. 19th IEEE Infocom*, 26 Mar 2000.
- [8] *Operations Application Messages-Language for Operations Application Messages*, Bellcore standard GR-831.
- [9] *FT-2000 OC-48 Lightwave System Releases 8.1, User/Service Manual* (Lucent Technologies, 2002).
- [10] *XQuery 1.0*, <http://www.w3.org/XML/Query>
- [11] H. Galhardas, *Data Cleaning Compilation*, <http://cosmos.inesc.pt/~hig/cleaning.html>
- [12] H. Galhardas, *Dirty Data*, <http://cosmos.inesc.pt/~hig/dirty.html>
- [13] S. Acharya, B. Gupta and P. Risbood, "Hitless Network Engineering of SONET Rings", in *Proc. IEEE Globecom*, 1 Dec 2003.
- [14] S. Acharya, B. Gupta and P. Risbood, "MobiPack: Optimal Hitless SONET Defragmentation in Near-Optimal Cost", in *Proc. 23rd IEEE Infocom*, 7 Mar 2004.