NetworkMD: Topology Inference and Failure Diagnosis in the Last Mile

Yun Mao
University of Pennsylvania
maoy@cis.upenn.edu

Hani Jamjoom
IBM T. J. Watson Research
jamjoom@us.ibm.com

Shu Tao
IBM T. J. Watson Research
shutao@us.ibm.com

Jonathan M. Smith
University of Pennsylvania
jms@cis.upenn.edu

ABSTRACT

Health monitoring, automated failure localization and diagnosis have all become critical to service providers of large distribution networks (*e.g.*, digital cable and fiber-to-the-home), due to the increases in scale and complexity of their offered services. Existing automated failure diagnosis solutions typically assume complete knowledge of network topology, which in practice is rarely available. The solution presented in this paper—Network Management and Diagnosis (NetworkMD)—is an automated failure diagnosis system that can infer failure groups based on historical failure data, and optionally geographical information. The inferred failure groups mirror missing topologies, and can be used to localize failures, diagnose root causes of problems, and detect misconfiguration in known topologies. NetworkMD uses an unsupervised learning algorithm based on non-negative matrix factorization (NMF) to infer failure groups. Using cable network as the primary example, we demonstrate the effectiveness of NetworkMD in both simulated settings and real environment using data collected from a commercial network serving hundreds of thousands of customers via thousands of intermediate network devices.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network topology*

General Terms

Algorithms, Measurement

Keywords

Failure Diagnosis, Network Topology Inference

1. INTRODUCTION

Service providers are offering bundled services to encourage consumers to use their high-speed data distribution networks. Adding these new customers and their services results in physical connectivity to support hundreds of thousands of devices as well as logical (or service) connectivity to support the evolving service offerings (e.g., Voice-over-IP and Video-on-Demand). There are significant network management challenges that arise from the combination of rapid growth and increased complexity of the network, particularly those associated with failures, which at this scale are almost certain to be present. This paper is focused on two challenges identified by multiple service providers we work with: *missing device status information* and *incomplete topology.* Cable network infrastructures are used as the primary examples in this paper, but the findings are equally applicable to other large-scale distribution networks, including IP networks, utility networks, etc.[1]

In a cable network, a single administrative area encompasses hundreds of thousands of end-customers (e.g., cable modems and VoIP phones) with thousands of intermediate distribution devices that operate on different protocol layers. Such devices include routers, Cable Modem Termination Systems (CMTSs), fiber nodes, repeaters, etc.

Consider a simplified cable network in Figure 1. A first challenge is that the Network Operations Center (NOC) cannot continuously monitor the status of all devices because either: (1) some devices are passive and therefore unresponsive to diagnostic packets or signals, or: (2) the cost of probing all devices is too high. An example device likely to be unresponsive is a physical layer repeater connecting cable modems and fiber nodes, and their statuses are rarely visible to the NOC. The computing time and resources required to continuously probe all cable modems in an administrative area are prohibitive, given the large size of the corresponding customer-base. Therefore, in practice, the NOC must rely on monitoring data limited to coarse-grained passive monitoring or infrequent active probes obtained from accessible devices to diagnose failures spanning the entire network. A second challenge is that in many cases, a complete network topology is not available. This typically happens because topology information is spread across disparate plant mapping applications, which may not be integrated with monitoring and management applications.[2] As a result, the NOC

[1]Our primary focus is on the *"last mile"* of the distribution infrastructure, where the logical topology is tree-like.
[2]We emphasize that typically this information is available.

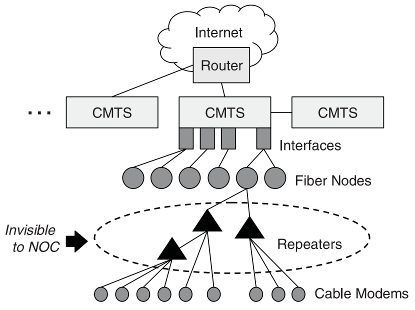


Figure 1: A typical cable network topology consisting of routers/switches, CMTSs, fiber nodes, repeaters, and cable modems, organized in a tree structure. Here, connectivity between end modems and fiber nodes is missing.

often needs to deal with an incomplete topology as in Figure 1, where the exact number of repeaters and their connections to fiber nodes and cable modems are unknown.

A desirable approach to address these challenges is one which *automatically* infers missing topologies in distribution infrastructures. To this end, we adopt the concept of *failure group* (FG)—a group of end components (i.e., cable modems) that are likely to share the same risk of failure. The failure of an FG dictates the failures of all its components. Failure groups are constructed by mining historical failure patterns and are shown to be good reflection of actual topological dependencies. While typically the missing topology cannot be fully discovered, we show how FGs are useful in failure diagnosis.

Specifically, we use FGs to help identifying the root cause of failures, even though failures may be caused by devices that are neither remotely measurable by the NOC, nor visible from the topology standpoint. In Figure 1 then, we are interested in identifying which repeater has failed based on the observation of health information of the higher- and lower-layer devices. Previous solutions to network failure diagnosis depend on the availability of complete topology (with possibly incorrect) information [15, 14, 22], which is used to understand the dependency between network devices (e.g., routers) and observed failures (e.g., IP link failures). Unlike these works, our study focuses on the problem of failure diagnosis with *incomplete information* about device status and network topology. The idea is that if we can infer FGs that directly reflect the missing part of the topology, we can then use FGs to localize failures, diagnose root causes, and even detect misconfigurations in the existing topology. We demonstrate that this approach can provide meaningful assistance to failure diagnosis in cable networks.

The rest of this paper is organized as follows. Section 2 provides a formal statement of the problem. We then introduce the details of our approach in Section 3. In Section 4, we describe the implementation of the NetworkMD algorithms in a real operational environment. In Section 5, we evaluate the effectiveness of our algorithms using both simulated and real data. Related works are discussed in Section 6 and finally, Section 7 concludes the paper.

It is however (1) extremely expensive to integrate into every monitoring application, and (2) may not be updated frequently enough to capture the actual topology as customers constantly join and leave the network, causing invisible changes.

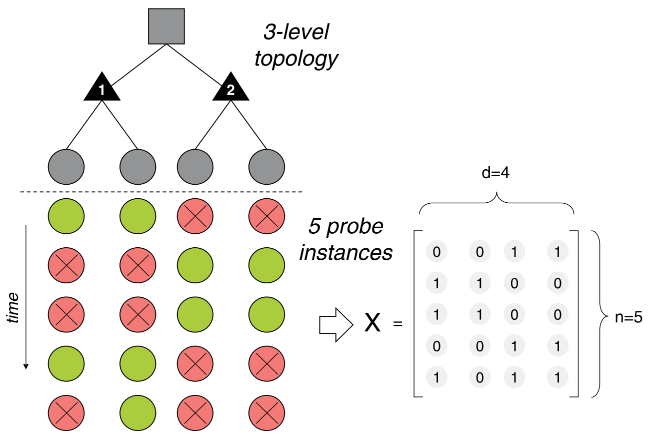


Figure 2: An example of the measurement process on a simple topology. Modems marked by an "X" indicate a failure.

2. PROBLEM STATEMENT

The primary information source to infer failure groups is the status of end-devices represented as a binary value, 1 if faulty, 0 if not. In cable networks, end-devices can be cable modems, set-top boxes, etc. As noted previously, the high overhead of probing makes it uncommon for management applications at the NOC to periodically check the status of all end-devices. Therefore, we do not assume the availability of complete status information for all devices in each measurement epoch. Rather, active probing is invoked only when a higher-level device (e.g., a CMTS interface) issues an alarm that indicates the occurrence of failure event. For instance, an alarm could be triggered if the number of *live* modems registered at a CMTS interface is less than a predefined minimum threshold.

The above measurement yields a *failure instance matrix* $X = [X_{ij}]_{n\times d}$ for each parent device (e.g., CMTS interface, node, etc), where n is the number of measurements (not necessarily obtained with a fixed period), and d is the number of end-devices probed in each measurement. For instance, X_{ij} represents the status of the j-th modem during the i-th measurement: $X_{ij} = 1$ indicates that the modem is faulty; $X_{ij} = 0$ means the modem is functioning.

In practice, the failure instance matrix collected by probes is not always accurate. This is caused by the facts that (1) probes are unreliable, hence may not always get responses; (2) monitored cable modems may be powered off when the probes are sent to them (so that these modems can be perceived as "faulty"). Furthermore, not all cable modem failures can be characterized by failure groups. In other words, probes may detect isolated cable modem failures, which are not caused by higher level device failures, hence should only be considered as measurement noise.

To illustrate the failure instance matrix, consider the simple 3-level topology shown in Figure 2. Active probes are launched from the root to all leaf nodes. The causes of the failures detected by the second and third probe are the failure of repeater 1. The failures detected by the first, fourth and fifth probes are contributed by the failure of repeater 2. Note that as perceived by the fifth probe, modem 1 is

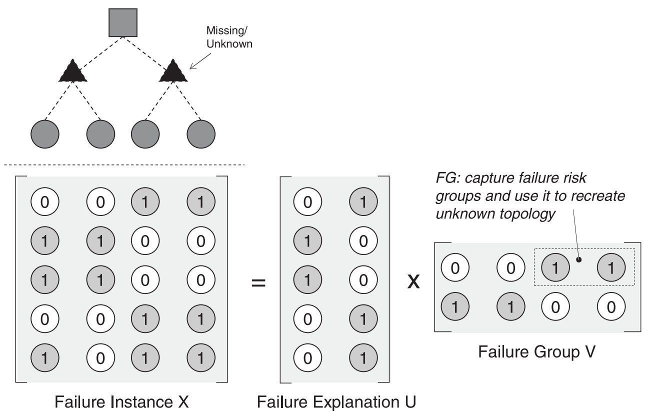


Figure 3: An example of decomposition of failure instance matrix X into failure explanation matrix U and failure group matrix V.

powered off by the user. Hence, it should be considered as an isolated failure, i.e., noise in the data set.

There is flexibility in choosing the level of network hierarchy at which the failure instance matrix is organized. For example, the matrix can group cable modem measurements at the CMTS interface level, yielding a small number of large instance matrices. Alternatively, the matrix can group measurements at the node level, yielding a large number of smaller instance matrices. The choice of grouping level depends on the target application, as well as the availability of data.

Given a failure instance matrix X, our goal is to identify the FGs and determine their failure statuses. Note that the number of FGs, r, is typically much smaller than the number of modems, d, due to the high-density tree topology of the network. Ideally, without any noise and measurement errors, the status of FGs and the association of cable modems to different FGs jointly determine X. Specifically, suppose we know *a priori* the compositions of all FGs and their statuses during the measurement, we can construct two binary matrices, a *failure explanation matrix*, $U = [U_{ij}]_{n\times r}$, and a *failure group matrix*, $V = [V_{ij}]_{r\times d}$. Each row of V represents an FG: $V_{ij} = 1$ iff the j-th modem is associated with the i-th FG, $V_{ij} = 0$ otherwise. As exemplified in Figure 3, each column of U represents the status of an FG: $U_{ij} = 1$ iff the j-th FG fails during the i-th round of measurement, $U_{ij} = 0$ otherwise. Thus, the product of the two binary matrices should equal the original failure instance matrix X:

$$X = U \times V \tag{1}$$

In practice, due to the noise in the data set, X might not exactly equal $U \times V$. Therefore, we need to find U and V such that they best represent the actual causes of the failures, despite the presence of noise.

3. APPROACH

We begin by describing the algorithm to find the failure group and failure explanation matrices. Figure 4 provides a high-level overview of the key steps. Section 3.1 describes the basic algorithm, which assumes the number of failure groups is known *a priori* and that the failure groups are non-overlapping. Section 3.2 extends the algorithm to cases

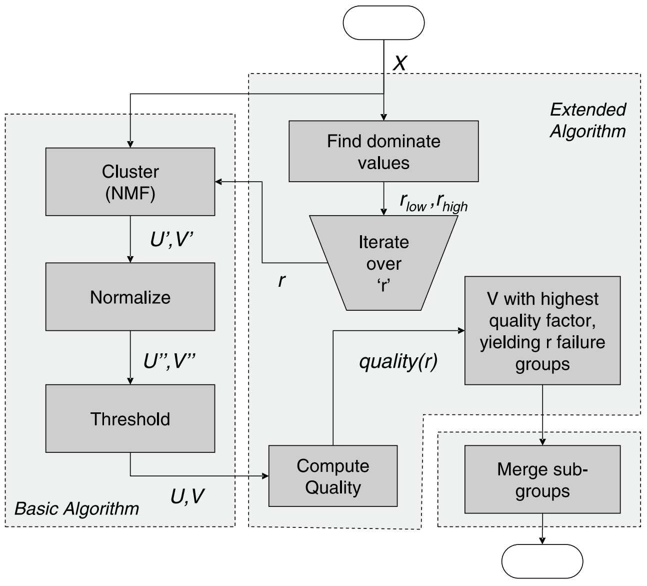


Figure 4: Outline of NetworkMD algorithm

where the number of failure groups is unknown. Finally, Section 3.3, extends the algorithm to handle more complex topology configurations. An alternative approach based on the k-means algorithm is given in Section 3.4 for comparison purposes.

3.1 The Basic Algorithm

For the basic algorithm, three assumptions are made. First, the failure instance matrix X is complete without missing values. Second, the target number of failure groups, r, is known. For example, the number of repeaters in the network is assumed to be known, but not the connections between cable modems and these repeaters. Third, the failure groups do not overlap, as in the case of cascaded topology. We will extend our study on cascaded topologies in Section 3.3.

We propose an algorithm based on the *Non-negative Matrix Factorization* (NMF) method [17]. NMF decomposes X into two non-negative real matrices $U'_{n\times r}$ and $V'_{r\times d}$, such that the *derived failure instance matrix*, $X' = [X'_{ij}] = U' \times V'$, is a good approximation of the original failure instance matrix X, i.e., NMF aims at minimizing the reconstruction error function:

$$\delta = ||X - X'||^2 = \sum_i \sum_j (X_{ij} - X'_{ij})^2 \tag{2}$$

The error function is minimized through an iterative process. Compared to general algorithms for minimizing functions, such as gradient descent or the Simplex Downhill method, the algorithm for NMF converges much faster. It requires no "magic number" tuning, such as choosing a step size in the Simplex Downhill method. The algorithm takes two random non-negative matrices $U' = [U'_{ij}]_{n\times r}$ and $V' = [V'_{ij}]_{r\times d}$ as input, and updates them in an alternating fashion. Specifically, in each iteration, we update U' and V' as:

$$\begin{aligned} U'_{ij} &\leftarrow U'_{ij}\frac{(XV'^T)_{ij}}{(U'V'V'^T)_{ij}} \\ V'_{ij} &\leftarrow V'_{ij}\frac{(U'^T X)_{ij}}{(U'^T U'V')_{ij}} \end{aligned}$$

The details of the iterated updating algorithm can be found in [17], in which Lee et. al. proved that these update rules converge monotonically to stationary points of the error function, Eq. (2). Based on our experience, several hundred iterations suffice for convergence to a local minimum.

Typically, the results of NMF are real values, which are inconsistent with the definition of $[U_{ij}]$ and $[V_{ij}]$, as U_{ij} and V_{ij} should both be binary values. To obtain a binary decomposition, we must convert U' and V' to binary matrices. Note the actual values in either U' or V' alone are not representative since one could multiply all elements in any column of U' by a constant and divide all elements in the corresponding row of V' by the same constant without affecting the validity of the decomposition. Therefore we cannot simply normalize them individually. To ensure consistent binary decomposition, we first normalize both matrices. This step is formalized by the following equations:

$$U''_{ij} = \frac{U'_{ij}}{\max_k U'_{kj}} \tag{3}$$

$$V''_{ij} = \frac{V'_{ij}}{\max_k U'_{ki}} \tag{4}$$

Simply put, the normalization factors are decided by the maximum values of the column vectors in U'.

After normalization, one can consider U''_{ij} as the confidence factor that indicates whether the j-th FG failed during the i-th probing, with 1 reflecting the highest likelihood and 0 reflecting the least likelihood. Similarly, V''_{ij} is the confidence factor of how likely device j would be part of FG i.

We then apply the following threshold-based algorithm to obtain the final binary matrices U and V:

$U_{ij} = 0$ iff $U''_{ij} < 0.5$; 1 otherwise;
$V_{ij} = 1$ iff the following three conditions are met:

$$V''_{ij} = \max_k V''_{kj} \tag{5}$$

$$V''_{ij} > 0.5 \tag{6}$$

$$V''_{ij} > 0.5 \times \max_k V''_{ik} \tag{7}$$

The threshold on U is self-explanatory. We impose three conditions on the threshold of V for the following reasons. Eq. (5) implies that a node can only be associated with one FG, which is the one of the maximum confidence factor V''_{ij}; Eq. (6) guarantees the absolute confidence factor is above 0.5; Eq. (7) indicates that a node is included in an FG only if the node's confidence factor is at least half of the confidence factor from the node which is most likely to be in the failure group.

Note that in the failure group matrix V, a node is not necessarily assigned to an FG. That is, it is possible that $\exists j, s.t. \forall i, V_{ij} = 0$. This is expected, especially for instance matrices with under-representative number of failures. For example, a node j would not belong to any FG if it never experienced a failure or if it failed independently from other nodes (which would have constituted the corresponding FG). Such behavior would lead to an all-zero column in the failure group matrix V.

We use the example shown in Figure 2 to demonstrate how the algorithm works. First we use NMF to decompose

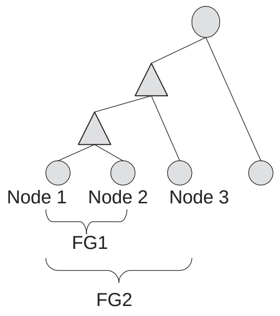


Figure 5: Cascaded topology with sub failure groups

X into two smaller matrices U' and V':

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \approx \begin{bmatrix} 0 & 0.57 \\ 1.28 & 0 \\ 1.28 & 0 \\ 0 & 0.57 \\ 0.56 & 0.59 \end{bmatrix} * \begin{bmatrix} 0.84 & 0.71 & 0 & 0 \\ 0.31 & 0 & 1.73 & 1.73 \end{bmatrix}$$

Since U' and V' are not normalized, looking at each value separately does not provide enough information to correctly determine the correct binary value. For example, both U'_{51} and U'_{52} are above 0.5, but the expected binary values should be $U_{51} = 0$ and $U_{52} = 1$ in the example. We apply the normalization algorithm and obtain:

$$U'' = \begin{bmatrix} 0 & 0.95 \\ 1.0 & 0 \\ 1.0 & 0 \\ 0 & 0.95 \\ 0.44 & 1.0 \end{bmatrix}, V'' = \begin{bmatrix} 1.07 & 0.91 & 0 & 0 \\ 0.18 & 0 & 1.03 & 1.03 \end{bmatrix}$$

Applying the threshold to U'' and V'', we get the final results:

$$U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

The above U and V exactly mirror the 'real' topology as depicted in Figure 2. Note that due to the noise in the measurement ($X_{51} = 1$ in particular), we also see some noise in U'' and V''. For example, the confidence factor of node-1 belonging to the second FG is $V''_{21} = 0.18$. However, since the failure observations are dominated by the failures of FGs, such noise can be easily identified and filtered out by the thresholds we set in Eqs. (5)-(7).

3.2 The Extended Algorithm

The basic algorithm assumes that the number of failure groups r is given. However, this assumption may not be realistic in practice. Therefore, we extend the basic algorithm to deal with the cases where r is not specified as an input parameter. Specifically, if we view the failure group inference problem as a clustering problem, then the problem of no prior knowledge of r is equivalent to a clustering problem without knowing the number of clusters. Hence, we develop the following procedure to identify FGs in this case.

The extended algorithm consists of three steps: (1) identify a range of possible values of r; (2) starting from the lower bound of r, apply the basic algorithm to find the failure group and failure explanation matrices and estimate its

```
function merge(U, V)
//$U$ is the n-by-r failure explanation matrix;
//$U_i$ is the $i$th column vector;
//$V$ is the r-by-d failure group matrix;
//$V_i$ is the $i$th row vector;
(n,r)=size(U); (r,d) = size(V);
for $i$=1 to $r$-1 {
  if ($\vec{U}_i != \vec{0}$)
  for j=i+1 to r {
    if($\vec{U}_i == \vec{U}_j$) {
      //FG$_i$ and FG$_j$ always fail at the same time,
      //merge them
      $U' = U_{1,..,j-1,j+1,..,r}$;
      $V' = V_{1,..,j-1,j+1,..,r}$;
      $V'_i = V_i \cup V_j$;
      return merge($U'$, $V'$);
    }elseif ($\vec{U}_i \supseteq \vec{U}_j$) {
      //FG$_i$ always fail when FG$_j$ fails
      $V_j = V_i \cup V_j$;
    }elseif ($\vec{U}_i \subseteq \vec{U}_j$) {
      //FG$_j$ always fail when FG$_i$ fails
      $V_i = V_i \cup V_j$;
    }
  }
}
return (U, V);
```

Figure 6: The algorithm to merge disjoint sub-failure groups.

error using the quality metric as in Eq. (3.2); (3) gradually increase r until the results satisfy certain criteria. By estimating the range of r first, we reduce the computation cost of unnecessary iterations on unrealistic guesses and the possibility of getting in local minima.

We estimate the range of r by finding the number of dominant singular values of X. According to Li [18], the number of clusters of a binary matrix X is close to the number of dominant singular values of X. Essentially, we try to look for a large gap between the singular values σ_r and σ_{r+1} of X. The lower bound and the upper bound of r is derived as follows:

$$r_{\text{low}} = \min\{i|\sigma_i < 10\sigma_{i+1}\}$$

$$r_{\text{high}} = \min\{i|\sigma_i < 2\sigma_{i+1}\}$$

After obtaining the range of r, we run our algorithm on each possibility of r from r_{low} to r_{high}. On each instance r_i, the quality of the result is calculated as follows:

$$\text{quality}(r_i) = \lambda_1 \frac{||X - U \times V||}{dn} + \lambda_2 \frac{||V - V''||}{r_i d}$$

That is, quality(r_i) is the quality metric of how good the estimation assuming there are r_i FGs. Smaller value of quality(r_i) indicates better quality. $\frac{||X-U\times V||}{dn}$ measures the binary reconstruction error to the original failure instance matrix per element: the larger the error, the worse the estimation quality; the second term $\frac{||V-V''||}{r_i d}$ represents the difference between the failure group matrices before and after the threshold step. The larger the difference, the more

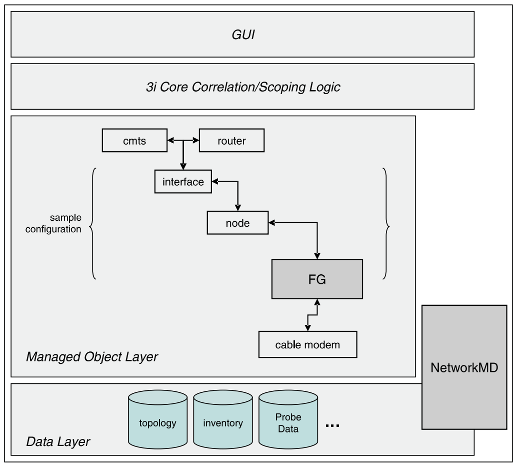


Figure 7: System architecture of 3i. The FGs are exposed to the user as virtual devices in the Managed Object Layer.

uncertainty introduced by the threshold, which leads to poor quality results. For example, $v''_{11} = 0.99$ gives us much more confidence than $v''_{11} = 0.51$ if in both cases the threshold algorithm yields $v_{11} = 1$. λ_1 and λ_2 are the parameters that determine how important the reconstruction error factor and threshold confidence factor weight relatively to each other. Our empirical study shows that $\lambda_1 = \lambda_2 = 1$ gives a good balance. Note that quality(r_i) also depends on the initial random matrices that used in NMF. We repeat the algorithm multiple times and choose the instance with the best quality score. Finally, we choose

$$r = \underset{r_{\text{low}} \le r_i \le r_{\text{high}}}{\text{argmin}} \text{quality}(r_i),$$

which is the number of FGs that yields the best quality score within its estimated range.

3.3 Dealing with Cascaded Topology

If part of the missing topology resembles a cascaded topology, some failure groups may be subsets of other larger failure groups. For example, the topology in Figure 5 has node 1 and 2 in FG_1, and node 1, 2 and 3 in FG_2. In this case, the algorithm we described above will split FG_2 and yield two groups: FG_1={1,2} and FG'_2={3}. Such cases can be detected by examining the corresponding failure explanation matrix U. It can been seen that whenever FG'_2 fails, FG_1 will fail simultaneously. Therefore, we can merge FG_1 and FG'_2 into $FG_2 = FG_1 \cup FG'_2 = \{1, 2, 3\}$ and leave FG_1 as is, which mirrors the topology more accurately. The algorithm to merge those failure groups is shown in Figure 6.

3.4 Alternative Algorithms

Looking at the problem from another perspective, we can view the FG association as a clustering problem. Given the n-by-d failure instance matrix X, each node has a feature vector—the corresponding column vector in X. Each feature vector has n dimensions. A standard clustering algorithm can divide the nodes into several disjoint clusters based on their features, so that nodes from the same clusters belong to the same FG.

Hence, a good alternative is to use standard clustering

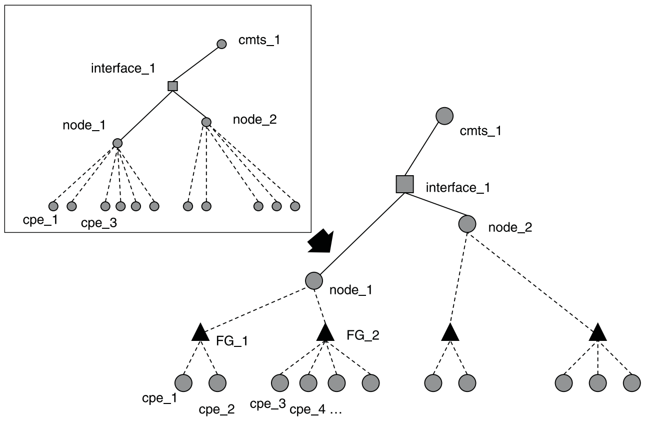


Figure 8: Topology transformation after FG discovery. The new topology contains the FGs as a layer between the customer premise equipment (CPE) layer and the fiber node layer.

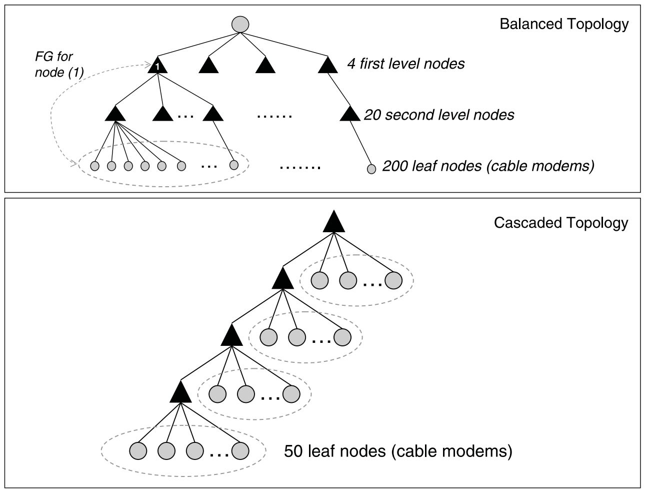


Figure 9: Simulated topology

algorithms, such as k-means [1], to derive FG association. After getting the FG composition, other failure diagnosis algorithms such as greedy min-set-cover [15] can be used to further decide which FGs are faulty based on the measurement result. However, one still faces the problem of not knowing the FG number and having sub-failure groups. The same techniques we proposed in section 3.2 and 3.3 still apply.

One advantage of the clustering algorithm is that the feature vector is extensible to information beyond failures. For example, in some applications, nodes that are physically closer to each other are more likely to be in the same FGs. If the geographic location of every node is known, then for node i, its feature vector can be extended from $\{X_{1i}, X_{2i}, \cdots, X_{ni}\}$ to $\{X_{1i}, X_{2i}, \cdots, X_{ni}, \lambda \cdot \mathrm{x_i}, \lambda \cdot \mathrm{y_i}\}$. Here (x_i, y_i) is the geographic coordinate of node i. λ is an adjustable weight parameter to indicate how important the geographic information is in the feature vector. However, as we shall see, these weight parameters usually take on some empirical values and could be hard to tune in practice.

4. IMPLEMENTATION

NetworkMD is implemented as a plug-in for a monitoring framework called 3i—Integrated Infrastructure Intelligence [12]. As a plug-in, NetworkMD has direct access to 3i's normalized monitoring data stored in a MySQL database. Upon completion, NetworkMD exposes its FGs as virtual devices that a user can interact with in 3i.

More specifically, 3i uses a device virtualization layer, called *Managed Object Bridge (MOB)*. As shown in Figure 7, a MOB instance can be a CMTS, an interface, or a cable modem, etc. While outside the scope of this paper, the MOB layer is intended to simplify building rich user-centric management applications. Furthermore, the MOB layer guarantees high-scalability through efficient management of disconnected graphs, with each device representing a node in the graph. NetworkMD uses the MOB API to *insert* its results as a virtual layer within the monitored topology. From a user's perspective, the FGs appear as new devices that can bring additional insight in the failure diagnosis process. Figure 8 shows how the results of NetworkMD are integrated with the real topology.

To ensure scalability, NetworkMD was implemented as two pieces. The first is PHP-based, which implements the MOB API. The second is a C-based PHP extension that performs matrix factorization. Implementing matrix factorization with compiled C code was necessary as our initial implementation, which was fully written in PHP, was inefficient in performing such periodic computation. After each round of FG inference, NetworkMD caches the results into a MySQL database.

5. EVALUATION

The algorithm has been validated using both simulation and experimental data collected from a real network setting. As mentioned before, the status information collected from cable modems is not always accurate. We thus focus the evaluation on both the accuracy of topology inference and the robustness in the presence of noise.

5.1 Simulation

A simulator was constructed, comprising about 1400 lines of MATLAB code. In the simulation, we use two representative topologies to study the performance of NetworkMD: the *balanced topology* and the *cascaded topology* as shown in Figure 9. In both cases, the simulation focuses on a single "unknown" repeater layer, with all parent-nodes in the simulated topology belonging to that layer. In our study, we only consider a partial topology associated with a single fiber node, because the full topology of a cable network can be viewed as the composition of several disjoint subgraphs, each associated with a fiber node and can be inferred independently using our algorithm.

That said, one can imagine the leaf nodes as cable modems and the intermediate nodes as the repeaters. The balanced topology is a 3-level tree, where the first level contains 4 nodes, forming the four major FGs that we want to identify. The second and third level of the tree have 20 and 200 nodes respectively. The number of nodes are well balanced across different branches. The cascaded topology, on the other hand, is not a balanced tree. On each level, there are 50 leaf nodes, plus one special node that connects the current and next level. These two topologies represent the two most common scenarios in practical cable networks. Our goal is then to identify the major FGs in the topologies, as identified by the circles in Figure 9.

Each time epoch is characterized by one or more failure incidents. In our simulation, we distinguish between *fail-stop* and *perceived failure*. A fail-stop is one where the node has experienced a software or hardware failure and can no longer function. In contrast, a perceived failure is one where an external monitoring application perceives that the node as faulty. When a parent node fail-stops, its child nodes will be perceived as faulty too. In our study, we simulate fail-stops, but also account perceived failures as input to infer the FGs.

We assume that nodes fail-stop independently and assign to the nodes at each level a certain failure probability: p_r for repeaters (or parent nodes) and p_m for modems (or leaf nodes). Typically, $p_r < p_m$. This is consistent with our observations from actual cable networks, where devices at the higher levels are less likely to fail. Note that p_m is used to simulate isolated cable modem failures, which are considered as noise in the input to our algorithm. However, as will be shown, the algorithm can still correctly identify the FGs based, despite the existence of such noise.

Our simulator also captures the typical way active probing is implemented in monitoring applications, where a certain threshold (e.g., the percentage of offline modems registered to an interface) must be reached before probes are launched to all end-devices. We simulate this behavior by recording the modem statuses *iff* the total failure ratio among all of the modems are above this threshold. When the threshold is exceeded, we record the failed modems as well as the failed upstream repeaters[3].

To evaluate the accuracy of the inferred FGs, we compute two metrics:

- *false positive ratio* R_p: A false positive occurs when two nodes are not in the same failure group described by the real topology, but the algorithm places them in the same group. R_p is the number of such node pairs divided by the number of all possible node pairs.

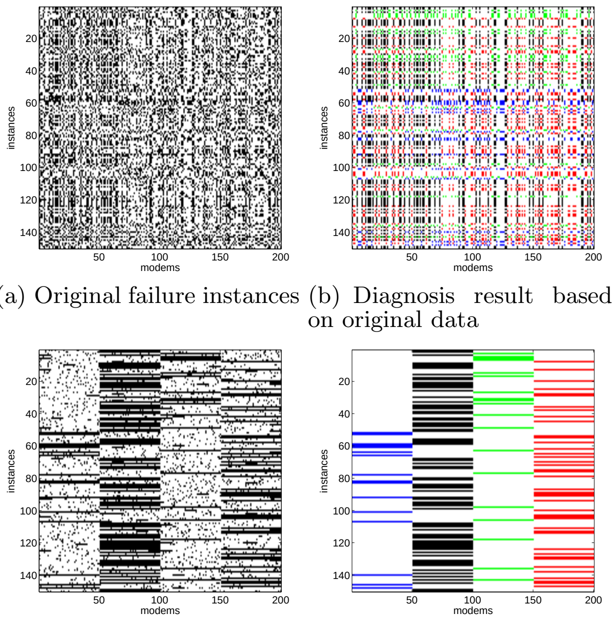
- *false negative ratio* R_n: A false negative occurs when two nodes that are in the same failure group, but the algorithm places them in different groups. R_n is the ratio of the number of such node pairs to the number of all possible node pairs.

We use *both* metrics to evaluate the effectiveness of our algorithm, with zero being the ideal value for both metrics. Note that looking at only R_p or R_n can be misleading. For example, an algorithm that puts every modem into an FG would have a zero false negative ratio, but the result is meaningless.

We start the simulation using a simple case study to demonstrate how NetworkMD performs in a balanced topology. We then study how different network topologies, numbers of observed failure instances, and measurement noises affect the performance of NetworkMD. We also compare the NMF-based algorithm against the one based on the *k-means* method.

5.1.1 *A Case Study on Balanced Topology*

[3]It is worth noting that repeater statuses are invisible to our inference algorithm and are used for its validation only.



(a) Original failure instances (b) Diagnosis result based on original data

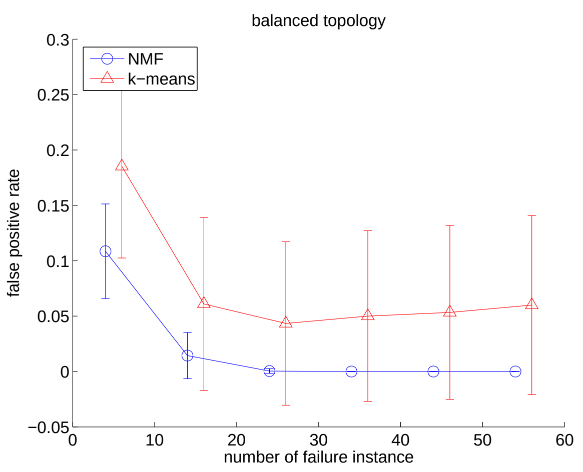
(c) Reordered failure instances (d) Reordered diagnosis result

Figure 10: Simulating FG inference on a two-level balanced topology

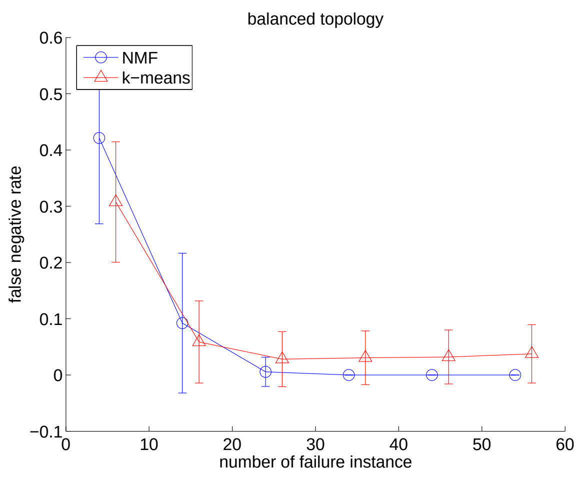
We first use the balanced topology to illustrate how NetworkMD works. We simulate a topology of 200 cable modems in a randomly generated balanced tree topology similar to that of Figure 9. Both level-1 and level-2 nodes are hidden from our algorithm. Here we show how we can identify level-1 nodes as major FGs. The algorithm has no prior knowledge of either the topology or the numbers of nodes on level 1 and 2.

In this simulation, we set the modem failure probability to $p_m = 0.1$ (note that p_m simulates noise) and set the repeater failure probability to $p_r = 0.5$. When the ratio of failed leaf nodes exceeds 25%, active probes are simulated and the statuses of all leaf nodes are collected as a measurement sample. We run the simulation until 150 such failure instances are observed.

Figure 10(a) shows the simulated failure matrix. A black dot in the figure indicates a failure observed on a modem in a probing instance. Note that the failure groups associated with the 4 level-1 nodes are hardly visible, unless we re-order the columns according to the modems' association with these 4 nodes (see Figure 10(c)). Using the original failure matrix as input, our algorithm infers 4 failure groups and computes the derived failure matrix (X'). The latter is shown in Figure 10(b) with each FG marked with a different color. If we reorder the columns in Figure 10(b) based on the modems' association to the derived FGs as in Figure 10(d), we can see the algorithm successfully filtered out the noise and accurately identified the dominant FGs represented by the level-1 nodes, i.e., $R_p = R_n = 0\%$. Note that despite the existence of noise caused by random cable modem failures, high level nodes have a bigger impact on leaf nodes if they fail. This explains why only the 4 FGs associated with the 4 level-1 nodes are identified.

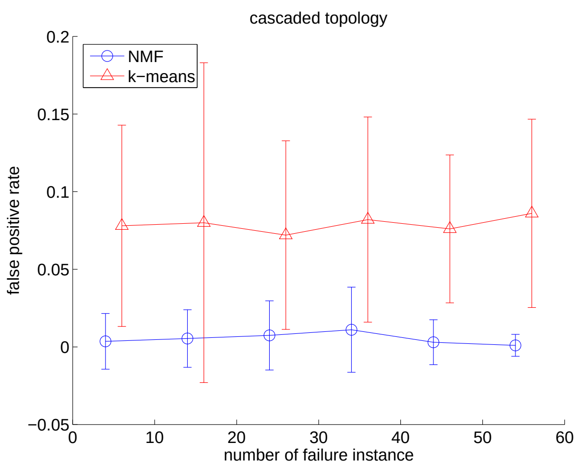


(a) False positive rate comparison

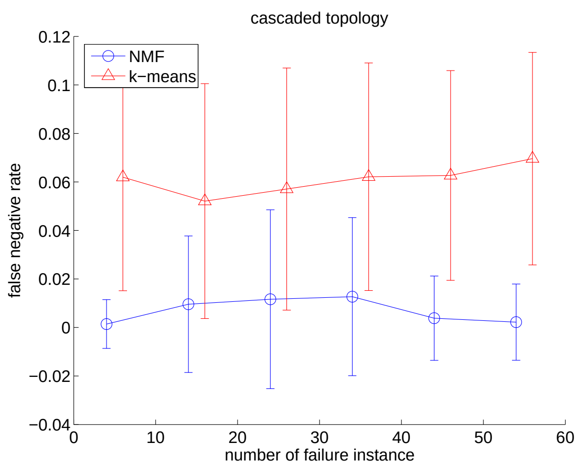


(b) False negative rate comparison

Figure 11: Comparing the effect of sample size on a balanced topology



(a) False positive rate comparison



(b) False negative rate comparison

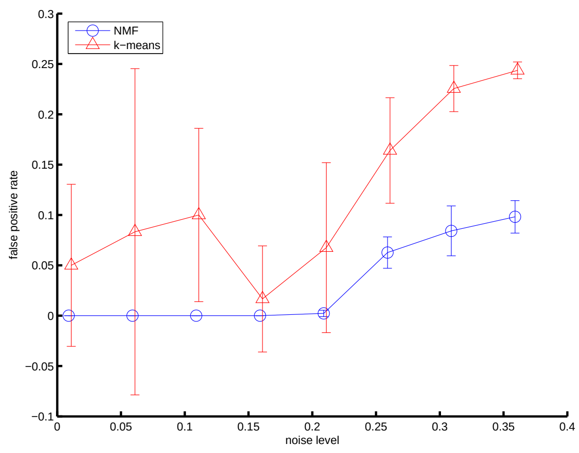
Figure 12: Comparing the effect of sample size on a cascaded topology

5.1.2 The Effect of Sample Size

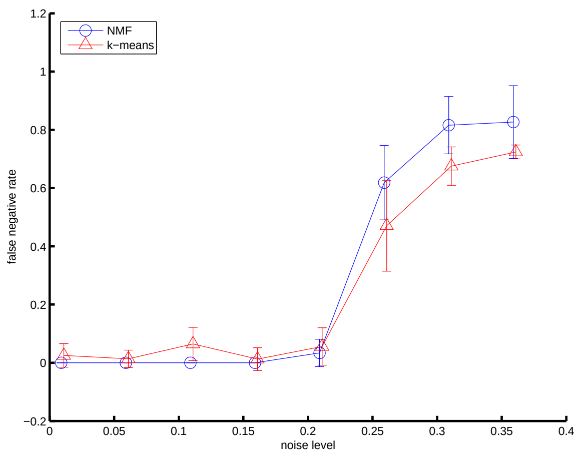
We vary the number of observed failure instances and study its impact on the performance of NetworkMD. In the simulation, we set the modem failure probability to $p_m = 0.1$, and the repeater failure probability to $p_r = 0.05$. We vary the number of observed failure instances (hence the number of measurement samples) from 30 to 250 and repeat each test 50 times.

Figure 11 compares the average false positive and false negative rates, as well as their standard deviation (shown as error bar), for the NMF and k-means algorithm, in a balanced topology. As the figure shows, the NMF-based algorithm has a lower false positive rate on average, and its standard deviation reduces to 0 when the number of samples is more than 30. That is, in the balanced topology, the NMF-based algorithm can always find the correct FGs given enough failure samples. However, having more samples does not necessarily help improve the accuracy of the k-means algorithm. While its average false positive rate decreases with the increasing number of failure samples, its standard deviation remains high. Therefore, even with a sufficiently large number of failure samples, the k-means algorithm cannot guarantee correct identification of all FGs in the balanced topology.

Similar to Figure 11, Figure 12 compares the false positive and false negative rates for the two algorithms in a cascaded topology. In this case, the average false positive and false negative rates of the NMF-based algorithm are very close to 0, regardless of the number of failure samples, and are much less than the rates of the k-means algorithm. Note that when the number of failure samples is small, the results inferred by the NMF-based algorithm in a cascaded topology is better than that inferred in a balanced topology. This is because in cascaded topologies those dominant repeaters are more likely to cause failures observed by the monitoring system, and therefore more likely to be identified. When more samples are observed, the effect of noise decreases to a reasonable level such that the accuracy of inference improves. When enough samples are collected, both algorithms give better



(a) False positive rate comparison



(b) False negative rate comparison

Figure 13: Effect of noise on a balanced topology

results in balanced topology than in cascaded topology since regularity is beneficial for our clustering-based algorithms.

5.1.3 *The Effect of Noise*

As noted earlier, individual modem failures are considered noise in the measurement data. To study the impact of noise, we simulate different noise levels in the balanced topology. The failure probability of repeater is set to p_r=0.01. We vary the failure probability of cable modem, p_m, from 0.01 to 0.36, and compare in Figure 13 the ratios R_p and R_n for NetworkMD, when it is equipped with NMF and k-means algorithms, respectively. The tests are repeated 10 times and we again show both the average value of R_p and R_n and their standard deviations. It can be seen that when $p_m \leq 0.21$, the NMF algorithm has false positive and false negative rates close to 0. Both NMF and k-means algorithms have higher false negative rate when the noise level p_r is higher than 0.25 (which is unlikely to be observed in practice). However, the false positive ratio of NMF algorithms is almost always less than 0.1, and has smaller standard deviation than the k-means algorithm.

5.1.4 *Simulation with Real Topology*

We extract a topology from a real cable network that we have monitored. The topology has a tree structure with two levels, consisting of roughly 3000 leaf nodes (i.e. cable modems) and 24 FGs (i.e. repeaters). The topology tree is not balanced. In the simulation, we vary the number of observed failure instances and study the impact on the performance of NetworkMD. We set the modem (leaf node) failure probability to $p_m = 0.1$, and the repeater (first-level node) failure probability to $p_r = 0.05$. We vary the number of observed instances from 30 to 250 and repeat each test 50 times.

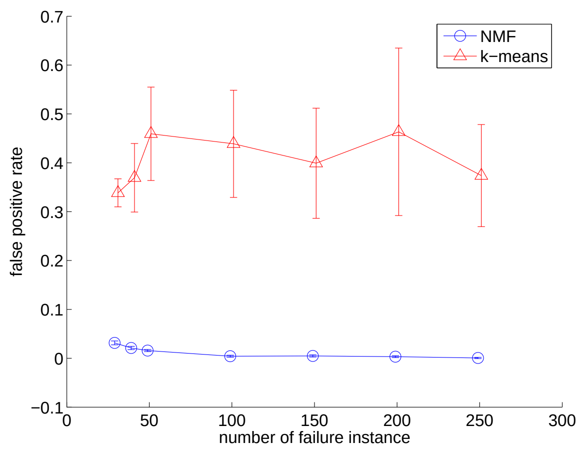
Figure 14 plots the average false positive and false negative rates when the NMF and the k-means algorithms are used in the extracted topology. As the figure shows, NMF algorithm always has a close-to-zero false positive rate. For the k-means algorithm , having more samples does not necessarily decrease its false positive rate. In terms of false negative rate, it is decreasing for NMF algorithm when more failure instances are provided as input. However, when there are more than 250 failure instances, both the NMF and the k-means algorithms provide satisfactory results.

5.2 Evaluations based on Real Datasets

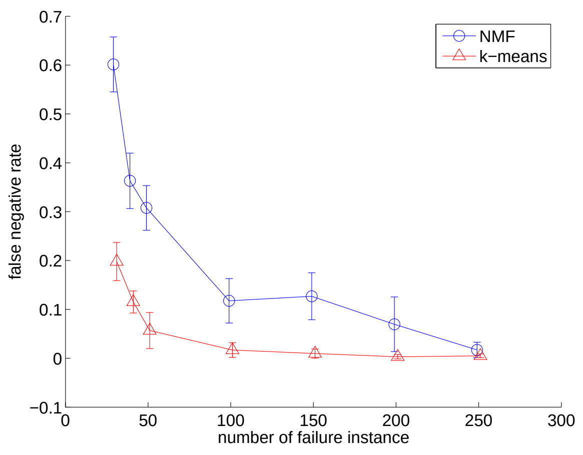
We have collected a set of monitoring data from a large cable provider that serves hundreds of thousands of customers and has a topology with thousands of intermediate network devices. The data contains detailed status report for all CMTSs, CMTS interfaces, and most of the underlying fiber nodes. Every CMTS interface has a register to record how many cable modems are physically connected to it, and how many of them are online. We launch active probes to all modems within a CMTS interface when an alarm is triggered by the interface, indicating that more than 15% of the modems are offline. Recall from the topology in Figure 1 that only the CMTSs, CMTS interfaces, and the underlying fiber nodes are visible to the NOC. Therefore, even though we can use NetworkMD to infer the missing topology between fiber nodes and cable modems, we do not have the actual connectivity information of those repeaters to validate our results. Such validation requires actual deployment of our system and access to failure incident records.

Because of the above limitations, we use the following methodology to validate our approach. We hide the connectivity information between the fiber nodes to the CMTS, and let NetworkMD infer the failure groups under the entire CMTS. Conceivably, the failures from those repeaters have less impact to the entire CMTS and should be considered as noise in the system. Hence, the identified major FGs should roughly match the composition of CMTS interfaces and fiber nodes, which will lead to more modem failures if any one of them fails.

We used the measurement data collected from a number CMTSs to validate our method. Here, we report the results for one of them. As shown by the measurement data, the specific CMTS has 3404 cable modems attached to it. During our monitoring process, we captured 53 failure instances (i.e., with 15% or more modems appeared offline for at least one of its interfaces). We found that 1933 cable modems have never been observed to be faulty or offline. These modems are excluded from our analysis. Among the 53 failure instances, 20 of them are identical to the previous



(a) false positive rate comparison



(b) false negative rate comparison

Figure 14: Effect of sample size on a topology extracted from a real data set

observation instances. These duplicate failure instances provide no additional information for FG identification, hence are also not considered in our study. As a result, our failure instance matrix X has 33×1471 elements.

Using X as input, we apply both NMF and k-means algorithms to identify FGs, and validate the results against the actual topology. For the k-means algorithm augmented with geographic information, we set the parameter λ to 0.05, which is a value we set to yield the best results with this data set. The results are summarized in Table 1. We see that the results derived by the NMF algorithm have similar accuracy compared those derived by k-means algorithm with geographic information, although false negative rate is slightly lower for the former. Note that without the geographic information, the k-means algorithm cannot converge in this case; thus FGs cannot be identified. The non-convergence issue is a major drawback of the k-means algorithm, especially when it is applied in high-dimensional data space. On the other hand, the k-means algorithm with geographic information requires careful tuning of parameter λ to balance the weight between failure observation and geographic information, which is nontrivial in practice. Hence, we conclude that the NMF-based algorithm is more suitable for a practical NetworkMD system.

We note that the false positive and false negative ratios are higher than those observed in our simulation. There are several potential reasons for that, aside from the effect of noise in the real data set.

- *Insufficient failure instances:* Many interfaces rarely failed during the monitoring period. As a result, it is difficult for our algorithm to discover the FGs associated with these interfaces.

- *Failure correlation:* We found that in the data set, there were many cases in which several interfaces always failed together. Since our algorithm is solely based on the failure pattern recognition, it would place the cable modems attached to those interfaces into the same failure group, which increases the false positive rate (when compared the derived FGs to the "ground truth").

- *Non-random failure noise:* As noted earlier, the ground truth of FG association is determined at the interface level. However, in the reality, repeaters under interfaces may also fail. During our monitoring period, although some interface never failed, a subset of the cable modems under certain repeater may failed together due to the repeater's failure. Such failure patterns will result in higher false negative rate, because our algorithm will not consider those never failed modems and the failed ones (although they are under the same interface) to be in the same failure group.

- *Misconfigurations:* It is possible that in the topology, the modems are mistakenly connected to CMTS interfaces. Such misconfigurations in the "ground truth" can lead to both higher false positive rate and higher false negative rate.

We manually examined the data set. Among the above 4 possible causes, we found the first one was the most prominent cause of inaccuracy. For example, if we simply merge the interfaces that hardly failed into one FG, the false positive ratio of NetworkMD using NMF algorithm would reduce to below 5%. The discrepancy between the inferred FGs and the real topology also indicates the possibility of misconfiguration in our topology. We have not yet confirmed this with the cable provider.

5.3 Modem Failure Estimation

As an application of NetworkMD, we demonstrate how to reduce the overhead of active probing by failure estimation. In the active probing phase, we allow a CMTS to send probes to only a subset of the modems to which it connects. Reducing the probe messages makes the whole monitoring infrastructure more scalable. With sampled probing, we estimate the status of all modems by leveraging the correlation of the modem failure pattern, or more specifically, the FG association identified by NetworkMD.

The failure estimation involves four steps. First, NetworkMD monitors all modems to derive the composition of FGs. Next, after the FG association is obtained, the system randomly select a subset of modems to probe when a

algorithm	false positive rate	false negative rate
NMF	0.09	0.133
k-means with geo	0.106	0.249
k-means without geo	N/A	N/A

Table 1: accuracy of NetworkMD in a real cable network.

failure event is detected[4]. Third, we apply the greedy min-set-cover algorithm proposed in [15] to identify which FGs are responsible for the failures. Finally, we use the combination of the failure explanation and FG composition to estimate the statuses of the modems that are not probed.

To evaluate the effectiveness of the above failure estimation procedure, we again use the data set described in section 5.2. We first partition failure instance matrix X into a training data set and a testing data set. The training data set is used to derive the FG composition (as in the first step). Then, we emulate probing 20% of the modems based on the testing data set, and estimate the failure statuses of the remaining 80% of the modems. We compare three cases in this study and show the results in Figure 15. In the oracle case, the actual FGs (based on the topology) are known *a prior* so that the training stage is unnecessary. This is obviously unrealistic, but can serve as a benchmark. In the cases when the NMF and k-means algorithms are applied, we rely on these two algorithms to infer FGs respectively, using the training data set. For all three cases, we measure the error in estimating the status of unobserved modems. The estimation error is measured by the ratio of the number of mistakenly estimated modems to the number of total unobserved modems. As the figure shows, due to the noise in the data set, even with an oracle of the topology, we cannot achieve 0% estimation error rate. The estimation error of NMF-based method is as low as 5% when more than 70% of the failure instances are in the training data set. This is better than the results obtained using the k-means method, which has at least 7% estimation error no matter how much data was used in the training phase.

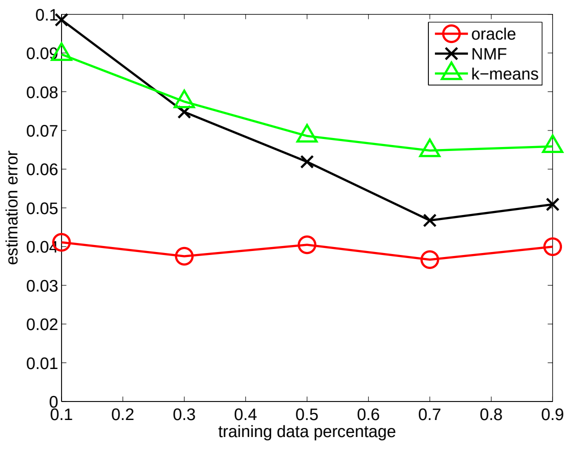


Figure 15: Study modem failure estimation by varying the training data set size

[4]Although a clever modem selection algorithm can make the estimation more effective, it is out of the scope of this paper.

6. RELATED WORK

Failure diagnosis is an important area in network management. There is a tremendous amount of work that studies this problem in IP, telephony, and cable networks. Commercial network failure management systems such as OpenView [10], SMARTS [7] and iGlass [11] provide frameworks for monitoring performance and handling failures. In particular, these systems are designed to interpolate with standard network management protocols, such as SNMP [2] or DOCSIS [13], and provide basic capabilities of rule-based correlation analysis. One limitation of such systems is that they can only discover failures of network devices that can be directly monitored.

There are a number of studies that looked into the problem of unobservable metrics [22, 15, 16, 23, 21]. For example, Steinder et. al. investigated an application of Bayesian reasoning using belief networks to locate faults in complex communication systems [22]. On the other hand, the *SCORE* system [15] and its later work [16] use an approach based on risk modeling. Risk modeling involves creation of a dependency relationship between observable events and potential causes. SCORE uses a greedy algorithm based on a min-set-cover technique to localize faulty devices whose statuses are hidden from the monitoring infrastructure. Kandula et. al. further studied the problem of noisy measurement and mis-configured risk group description. They proposed an algorithm called *Shrink* based on Bayesian networks with polynomial running time bound. However, all these systems assume that the relationship between the events and the causes are (mostly) known. In our problem setting, complete topology information is not required. The proposed solution does not assume the availability of such information, but instead infers the missing topology through failure group association.

Aside from the failure diagnosis and network management area, NetworkMD also relates to topology inference and discovery in the Internet [6, 3, 19, 9, 20]. These works usually require traceroute-like in-network probing or access to BGP routing table, neither of which is available in last-mile cable networks. A particular class of the research problems, called *network tomography* [8], describes a series of network inference and monitoring problems, such as traffic matrix estimation and topology identification, without in-network probing support. In particular, Coates et. al. investigated how to use end-to-end latency measurement to discover the network topology without performing traceroute [4, 5]. They proposed to use "sandwich" probes to measure delay difference and model the topology to maximize a penalized likelihood. In their work, a special Markov Chain Monte Carlo procedure was used to do maximized likelihood estimation. Unfortunately, these network tomography techniques cannot be directly applied in failure diagnosis in cable networks. For example, in [4], the end-to-end metrics are required to be *separable*, meaning a path metric can be decomposed into the metrics associated with the links comprising the path.

In cable network fault diagnosis, the metric of a path from a CMTS to a modem is a binary status indicator. It is unclear how to adapt their solutions to this problem setting. Unlike their work, the NetworkMD solution is designed to infer missing topologies based on binary failure status data.

7. CONCLUSIONS AND FUTURE WORK

NetworkMD (Network Management and Diagnosis) is an automated topology inference and failure diagnosis framework for last-mile distribution networks. It provides an unsupervised learning algorithm that creates failure groups based on end-to-end failure pattern measurement. The learning algorithm is based on non-negative matrix factorization (NMF) and further extended to deal with unknown number of FGs and cascaded topology. Besides the NMF-based algorithm, NetworkMD can also use a standard clustering algorithm (k-means) to infer FG association, while taking advantage of geographic information when available. These algorithms are not only applicable in cable network diagnosis, but can also be generalized to other networks with tree-like topologies. We have conducted extensive simulations and experiments with NetworkMD. Our evaluation shows its effectiveness in both simulated settings and for datasets collected from a commercial cable network.

Going forward, there are several interesting applications that can be built using NetworkMD. One such application is to combine the inferred FGs with geographic information of end-points to help physically localize failures (such as a cable cut). Another application can compare the inferred FGs to the known topology and flag inconsistencies as potential misconfigurations to the network operator. A third application can combine better sampling techniques with the inferred FGs to reduce the overhead of active probes, allowing service providers to respond faster to failures.

Acknowledgement

We thank Nikos Anerousis, Lawrence Saul, Jimeng Sun, Renata Teixeira, Kilian Weinberger and the anonymous reviewers for their helpful comments on the earlier versions of this paper.

8. REFERENCES

[1] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[2] J. Case, M. Fedor, M. Schoffstall, and J. Davin. RFC1157: Simple Network Management Protocol (SNMP). *IETF*, April 1990.

[3] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. Towards capturing representative AS-level internet topologies. *Computer Networks*, 44(6):737–755, 2004.

[4] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 11–20, 2002.

[5] M. Coates, M. Rabbat, and R. Nowak. Merging logical topologies using end-to-end measurements. In *Internet Measurment Conference (IMC)*, 2003.

[6] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. *SIGMETRICS Performance Evaluation Review*, 33(1):327–338, 2005.

[7] EMC. SMARTS. http://www.emc.com/products/software/smarts /smarts_family/.

[8] L. Gang, M. Coates, G. Liang, R. Nowak, and B. Yu. Internet Tomography: Recent Developments. *Statistical Science*, Mar 2004.

[9] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy. A systematic framework for unearthing the missing links: Measurements and impact. In *Proceedings of the 4th USENIX Symposium on Networked System Design and Implementation (NSDI)*, 2007.

[10] Hewlett-Packard. Management Software: HP OpenView. http://h20229.www2.hp.com/.

[11] iGlass. iGlass. http://www.iglass.net.

[12] H. Jamjoom, N. Anerousis, R. Jennings, and D. Saha. Service Assurance Process Re-Engineering Using Lacation-aware Infrastructure Intelligence. *the Tenth IFIP/IEEE International Symposium on Integrated Network Management*, May 2007.

[13] D. Jones and R. Woundy. RFC3256: The DOCSIS (Data-Over-Cable Service Interface Specifications) Device Class DHCP (Dynamic Host Configuration Protocol) Relay Agent Information Sub-option. *IETF*, April 2002.

[14] S. Kandula, D. Katabi, and J. P. Vasseur. Shrink: A tool for failure diagnosis in IP networks. In *Proc. of ACM SIGCOMM MineNet Workshop*, 2005.

[15] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *Proceedings of NSDI*, 2005.

[16] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Detection and Localization of Network Black Holes. In *Proceedings of IEEE Infocom*, May 2007.

[17] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 556–562, 2000.

[18] T. Li. A general model for clustering binary data. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 188–197, New York, NY, USA, 2005. ACM Press.

[19] Z. M. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz. Scalable and accurate identification of AS-level forwarding paths. In *Proceedings of IEEE Infocom*, Mar 2004.

[20] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level path inference. In *the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 339–349, 2005.

[21] P. Sebos, J. Yates, D. Rubenstein, and A. Greenberg. Effectiveness of shared risk link group auto-discovery in optical networks, 2002.

[22] M. Steinder and A. Sethi. Increasing robustness of fault localization through analysis of lost, spurious,

and positive symptoms. In *Proc. of IEEE INFOCOM, New York, NY, 2002.*, 2002.

[23] P. Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, and Z. Shi. Alarm correlation engine (ACE). In *Proceedings of Network Operations and Management Symposium'98*, Feb. 1998.