NetSearch: Googling Large-scale Network Management Data

Tongqing Qiu*, Zihui Ge†, Dan Pei‡, Jia Wang†, and Jun (Jim) Xu *
* College of Computing, Georgia Institute of Technology, Atlanta, GA
†AT&T Labs – Research, Florham Park, NJ
‡ Tsinghua University, Beijing, China

***Abstract*—In order to ensure the service quality, modern Internet Service Providers (ISPs) invest tremendously on their network monitoring and measurement infrastructure. Vast amount of network data, including device logs, alarms, and active/passive performance measurement across different network protocols and layers, are collected and stored for analysis. As network measurement grows in scale and sophistication, it becomes increasingly challenging to effectively "search" for the relevant information that best support the needs of network operations.**

In this paper, we look into techniques that have been widely applied in the information retrieval and search engine domain and explore their applicability in network management domain. We observe that unlike the textural information on the Internet, network data are typically annotated with time and location information, which can be further augmented using information based on network topology, protocol and service dependency. We design NetSearch, a system that pre-processes various network data sources on data ingestion, constructs index that matches both the network spatial hierarchy model and the inherent timing/textual information contained in the data, and efficiently retrieves the relevant information that network operators search for. Through case study, we demonstrate that NetSearch is an important capability for many critical network management functions such as complex impact analysis.

I. Introduction

Large IP networks are designed with the goal of providing high availability and Quality of Service while keeping the operational complexity and cost low. Meeting this goal requires the continuous measurement and monitoring of the network for detecting events and conditions that may threaten or compromise it. Towards this end, an Internet Service Provider (ISP) brings together a large amount of measurement data from across the network, including device log files, traps and alarms on different network layers, active or passive service performance measures, etc. Since manually inspecting these large data sets is extremely time consuming, they are typically streamed into a database management system (DBMS) such as Darkstar [1] and stored for later analysis.

While such a DBMS is intended to serve as a comprehensive, one-stop resource to consolidate information from all network data sources and make the data readily available to whoever needs it [2], it alone does not usually allow network operators to effectively *retrieve* the right data they need for network troubleshooting and diagnosis purposes. In particular, it allows for simple queries such as the ones based on the time window and/or router ID, but often cannot deal with more complicated ones that are used to efficiently identify the anomalies in the network such as those based on complex relationships (especially spatial relationships) among different messages[1] in the database. Therefore, additional search or information retrieval tools often need to be built on top of the DBMS to reduce the time it takes to troubleshoot network events. Although some specific commercial tools such as Lonix [3] and NetCool [4] are developed towards this end, they focus only on a small set of network messages concerning network faults. Because the syntax and relationships of messages are hard-coded in these tools to enable automatic parsing and understanding, they are inflexible in handling updates on the syntax and/or relationship models.

In this work, we present NetSearch, a search and information retrieval tool we have built and experimented on a tier-1 ISP network that works like a search engine over the large network measurement and monitoring data sets. However, its design is quite different from generic Web search engines because network data have complex relationships among them, although these relationships can indeed be learned as a part of network domain knowledge.

To support fast and accurate informational retrievals over the measurement and monitoring data, we need to parse and index them properly. Different from web search engines that deal with natural language documents [5], media types such as video and audio [6] and graphics [7], NetSearch focuses on parsing and indexing the measurement messages that describe network events. It is a very different (and hence nontrivial) task in two ways. First, each message contains both temporal and spatial information. The spatial information, in particular, is diverse and complicated. For instance, some messages describe the interface related information, while some others describe the protocol related information. Second, the relationships among different messages are complicated. In particular, multiple related measurement messages are typically generated by one or more related network events. These messages can come from different protocols (e.g. OSPF, BGP) across different network layers (e.g. layer-2, layer-3).

We expect that NetSearch will become a very handy information-retrieval tool for daily network operations. With NetSearch, network operators can search what they are interested in by describing the temporal and spatial aspects of network events and obtain the results instantly. NetSearch can extract the location information even if it is embedded in the network messages. By modeling the locations as a

[1]We call a data item in the database as a *message*. In the rest of this paper, the terms "data" and "messages" are exchangeably.

ISBN 978-3-901882-58-6

TABLE I. SYSLOG MESSAGES EXAMPLE

Vendor	Message timestamp	Router	Message-type/error-code	Detailed message
V1	2013-07-10 00:00:15	r1	LINEPROTO-5-UPDOWN	Line protocol on Interface Serial13/0.10/ 20:0, changed state to down
V1	2013-07-10 00:00:15	r5	LINK-3-UPDOWN	Interface Serial2/0.10/2:0, changed state to down
V1	2013-07-10 00:00:15	r8	SYS-1-CPURISINGTHRESHOLD	Threshold: Total CPU Utilization(Total/Intr): 95%/1%, Top 3 processes (Pid/Util): 2/71%, 8/6%, 7/3%
V2	2013-07-10 00:00:23	ra	SNMP-WARNING-linkDown	Interface 0/0/1 is not operational
V2	2013-07-10 00:00:24	rb	SVCMGR-MAJOR-sapPortStateChangeProcessed	The status of all affected SAPs on port 1/1/1 has been updated.

tree hierarchy, NetSearch can automatically identify the relationships among different messages even when they are originating from different data sources. The methodologies used in NetSearch are generic and therefore can be used in a wide range of network data sets. More importantly, it enables complex network management tasks, in which multiple data sets need to be considered simultaneously.

In summary, we make four major contributions in this paper:

1) We designed a tool called NetSearch for network operators to efficiently search all relevant network messages from different data sources.
2) We developed a systematic methodology to identify the location information embedded in the messages. We also formalized the relationship of different locations in network events.
3) We conducted the large-scale experiments on real network data collected from one tier-1 ISP network.
4) We demonstrated that NetSearch can significantly benefit the network management through real case study.

The rest of the paper is organized as follows. Section II overviews the syntax and semantics of data sets used in network management. Section III presents the overview of NetSearch system. Section IV describes the details of several key components of this system. Sections V and VI evaluate NetSearch through large-scale experiments and demonstrate its usage in case study. We briefly overviews related work in Section VII and conclude the paper in Section VIII.

II. DATA USED IN NETWORK MANAGEMENT

Many network management functions depend on a rapid and comprehensive view of the network events including device logs, traps, alarms, and operation tickets. We first describe several of the most commonly used network data sources.

- Logs: router SYSLOG is a classical example of textual based data source in which tremendous information regarding the nature of the network condition and the involved hardware component is embedded in the descriptive message body. Router config change log, network care tickets are similar data sources in this nature.
- Device alarms: most types of networking devices are capable of self-monitoring against local adversary conditions, generating traps or alarms once triggered. Such data typically are well structured and with detailed time and location information. We use Layer-1 SONET alarms as a representative of this type of data source.
- Output from network measurement systems: network monitoring systems, either through active measurements (e.g., ping test) or passive (e.g., traffic sniffing) ones, either on the data forwarding plane or on the control plane (routing), also produce tremendous information. Their outputs are likely in proprietary data format and are typically quite "coded", which are good for automated processing and not so for human readability.

We next describe one data source in each category in details to illustrate the data source diversity that NetSearch faces.

A. Router SYSLOG

Router SYSLOGs are the messages that routers generate to record the hardware and software conditions observed by them, such as link and protocol-related state changes (e.g., down or up), alarming environmental measurements (e.g., high voltage or temperature), and warning messages (e.g., triggered when BGP neighbors send more routes than the router is configured to allow). Although SYSLOG messages are intended primarily for tracking and debugging router software and hardware problems, they can be extremely valuable to network operators in managing networked services and troubleshooting network incidents.

SYSLOG messages are essentially free-form texts, the syntax and semantics of which vary among router vendors and router operating systems. Table I shows a few examples of SYSLOG messages from two router vendors. We can observe only a minimal structure in a SYSLOG message: (1) a timestamp indicating when the message is generated (with router clocks synchronized through NTP), (2) the identifier of the router that generates the message, (3) message type, also known as the error code, indicating the nature of the problem, and (4) detailed message information generated by the router OS.

The router identifier (2) is typically the IP address of the router loopback interface, which is converted into router name in our system for better human readability. The detailed message information (4) is quite ad hoc in nature. They are simply free-form texts "printf"-ed by router operating systems with detailed information such as the location, state/status, or measurement readings of an alarming condition embedded in them. For example, in Table I line 1, the `Serial13/0.10/20:0` part indicates the network interface at which the layer-2 line protocol has been impacted and the `down` part indicates the status of the line protocol.

TABLE II. SONET MONITORING MESSAGES EXAMPLE

Message timestamp	Port ID	Circuit ID	Router	Interface
2013-07-01 00:00:00	1s-1-1b-3	DHEC 32423	r1	T1_0/1/0:7
2013-07-01 01:00:00	ls-1-1b-1	DHEC 87907	r2	Serial4/0.8/2:0

TABLE III. OSPF MONITORING MESSAGES EXAMPLE

Message timestamp	Event ID	Event type	Router	IP
2013-07-10 00:00:15	604144	Link Down	r1	192.168.0.30
2013-07-10 00:00:16	234234	Link Cost Change	r2	192.168.0.34
2013-07-10 00:00:16	134323	Router Cost Out	r3	-

Router configuration tools usually allow network operators to specify a severity level for SYSLOG logging. In this study, we collect SYSLOGs at such "informational" level (usually the default setting). Depending on the network conditions, the amount of router SYSLOG messages in an operational network varies. In the large-scale ISP network (hundreds to thousands routers) that we study herein, there are typically up to millions of messages per day.

B. Layer-1 Device Alarms

In a network management system, a wide range of alarms are generated across different network layers. Different from higher level alarms, layer-1 (physical) alarms are typical triggered by hardware failure. In many cases, layer-1 alarms indicate the root causes of upper layer issues, and therefore, are considered as one critical measurement data source. Here we use SONET alarms as the typical example of layer-1 alarms.

Below the IP layer, the ISP uses a Synchronous Optical Networking (SONET) layer on top of fiber optics, which takes care of scheduling packets to be transported by way of Time Division Multiplexing (TDM) and handles rate multiplexing, traffic grooming, error monitoring and restoration. A number of alarms from the SONET layer can be recorded and timestamped by SONET equipments. Of these alarms, the Section Loss of Signal (SLOS) is the most critical one and is triggered when a failure occurs in the optical layer. When a failure occurs at the optical layer, a SLOS alarm is generated. A SONET equipment then waits for a small period of time, before reporting this failure to the IP layer. This is done to dampen out very short failures that disappear so as to avoid triggering route re-computation at the IP level [8].

Table II shows a basic example of SONET messages. Regarding the temporal information, SONET alarms include the timestamp of the events. Regarding the spatial information, raw SONET messages only contain port ID and circuit ID that have been assigned to the end of fiber link. Since our NetSearch system focuses on the IP layer, we preprocess the SONET messages at the time of data ingestion to map the underlying port ID and circuit ID to the associated layer-3 interface(s). Data preprocessing and annotation is an important step for NetSearch, as doing so greatly improves the readability for human operators to examine the data. There are also a number of counter values about the alarmed condition in the body of the alarm message (not shown in the table), presenting diagnostic information about the layer 1 issue. In the large-scale ISP network that we study in the paper, there are typically tens of thousands of such messages per day.

C. Control Plane Monitors

Besides networking devices, there exists many auxiliary systems that monitor the traffic flows and control plane messages. For instance, network operators pay considerable attention to the performance of the routing infrastructure by continuously monitoring message exchanges of routing protocols on the control plane [9], [10]. In this paper, we take Open Shortest Path First (OSPF) measurement data as one typical example. Other data sources such as BGP updates monitoring, active measurement based fault and performance problem detection (e.g., through ping test) would fit well in NetSearch.

OSPF is an adaptive routing protocol for IP networks. It uses a link state routing algorithm and operates within a single autonomous system. Tracking the changes to the OSPF neighbor states is key for managing an OSPF network. There are two approaches to do so: relying on SNMP MIBs and traps, or listening to Link State Advertisements (LSAs). The ISP uses the architecture in [10] for OSPF monitoring, in which multiple LSA Reflectors are deployed to capture LSAs from the network for both real-time and off-line analysis.

Raw LSAs archive is preprocessed and converted into high-level OSPF monitoring messages. Table III shows an example of these messages. Each OSPF message has a timestamp indicating the time when the event is observed by the OSPF monitor, a unique event ID serving also as a sequence number, a coded message type describing the nature of the OSPF event, such as link up/down or link weight change, and some detailed message information (e.g., the link weight before and after the event). Regarding the spatial information, each OSPF message contains the subnet IP address of the link where the event had taken place (a point-to-point link being a /30 subnet) and the router ID of the source end of the link to indicates the direction of the directed link – OSPF cost can be asymmetric. Or in some cases, if the event pertains to an entire router, e.g., when a router is costed out for service (all its outbound links are set at the maximum link weight), the subnet field is left empty. In the large-scale ISP network we study in the paper, there are typically tens of thousands of such messages per day.

D. Remark

We observe that all network messages share some minimal structure. First, each message describing a network event contains a timestamp to record the time when the event has been observed. Second, each message contains a source equipment ID to generally describe from where the event has been observed. In an IP centric system, the source equipment is typically a router. Knowing the source router however is often not enough to pinpoint the network event. The network messages do provide more detailed location information. While the exact location can be easily extracted from well-structured data sources such as OSPF Monitor and SONET alarms, in textual based logs, the locations can be embedded in the message body. NetSearch should extract the detailed location information from the messages. Moreover, the physical or logical component at which a network event took place is not in a flat structure. Some of them are meaningful at the router level (like CPU utilization in SYSLOG), some are at the IP link

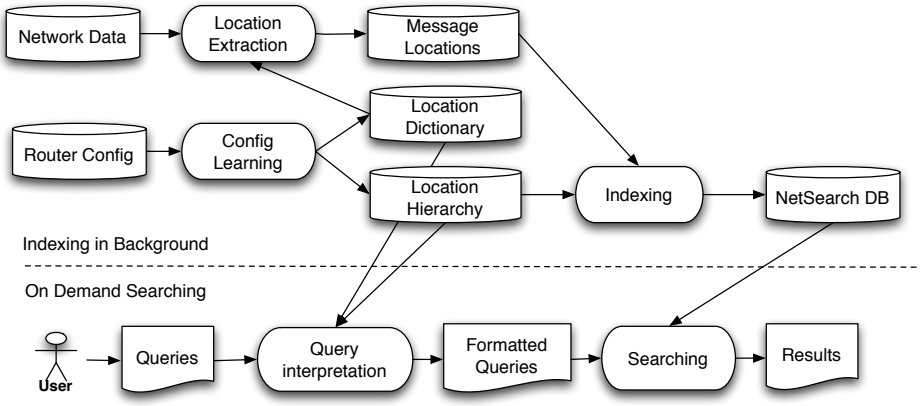

Fig. 1. NetSearch Architecture.

level (OSPF messages), and others are at a physical link level (e.g. Circuit ID in SONET), or a physical port, or a line card level. Therefore, in order to be able to search a queried spatial location in all relevant messages which could have various location types, we need to carefully model the relationships among different location types.

III. NetSearch System Overview

The goal of NetSearch tool is to search through vast volume of network data and retrieve relevant information in response to operator's query. Figure 1 shows the system architecture of NetSearch. It consists two parts: *indexing in background* and *on demand searching*.

A. Indexing in Background

NetSearch automatically and continuously acquires domain knowledge such as the detailed location information from raw messages, and then generates the indices for search purpose.

First, NetSearch learns the location information based on configuration files (we call this process as *configuration learning*). The first role of the location learning is to build a location "dictionary" for extracting the location part of each network message. The second role of location learning is to build the mapping between different locations, e.g., from an interface name to its IP address, and the hierarchical location relationship between interfaces, ports and linecards, and network topology such as the interfaces connecting two routers. These mappings and location relationships enable searching for relevant messages given a specific location description. We will discuss the configuration learning in details in Section IV-A.

Second, NetSearch automatically extract the location parts of messages. One primary method is based on the location dictionary generated by configuration learning process. However, not all locations appear in the configuration file. Therefore, we apply other heuristic methods. One complementary method is based on message context. For example, certain keywords like "interface" or "port" in the messages may indicate the location information. Another complementary method is based on domain knowledge of location string pattern. For example, an IP address has an obvious string pattern in dot-decimal notation. Since each individual method cannot achieve the high extraction accuracy, NetSearch combines these methods in an appropriate order. We will explain location extraction in details in Section IV-B.

Finally, we index raw messages into NetSearch database to serve online queries. More specifically, we divide our indexing into three independent parts: temporal indexing, spatial indexing and textual indexing. These three parts are the three key components of network data. The detailed methodology of indexing is discussed in Section IV-C.

B. On Demand Searching

Using the generated indices, NetSearch takes users' queries as input on demand, and output all relevant messages in a two-step process. The first step is to interpret users' queries. In this step, users may input the time window and location they are interested in. NetSearch allows some flexibility of location description. For example, users do not need to specify the type of the location (e.g. interface, linecard). NetSearch is able to interpret user queries automatically. The methodology of query interpretation are described in Section IV-D. The second step is to search all relevant messages. The searching step, especially location searching, is closely related to the location indexing. We will present detailed methodologies for location indexing and location searching together in Section IV-C.

IV. Methodology

In this section, we discuss several key methodologies used in NetSearch tool: configuration learning, location extraction, indexing and searching, and query interpretation. These key components in the system architecture are shown as squashed rectangles in Figure 1.

A. Configuration Learning

Each router is individually configured with its own router configuration file. In this study, we use the configuration files to generate both location dictionary and location hierarchy. The router configuration keeps changing as the network evolves over time. NetSearch system continuously updates the location dictionary and hierarchy generated based on router configurations.

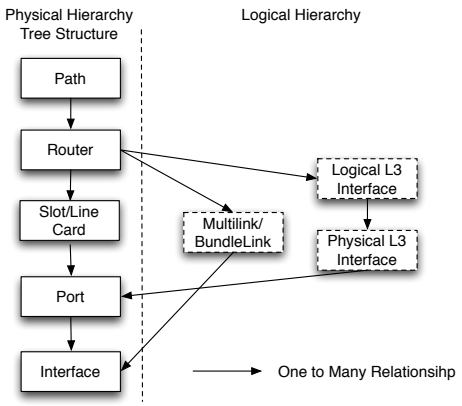

Fig. 2. Location hierarchy

1) Location Dictionary: We first extract the types of locations (e.g. interface, IP) and their actual values from the configuration files. We organize the information as a dictionary which contains a number of type-value pairs (e.g. Interface SERIAL2/0.7/11:0, IP 192.168.1.123). Using this location dictionary, NetSearch can understand the location semantic in the network data. Consequently, the dictionary is used to extract location information embedded in network messages (see in Section IV-B). Moreover, it can also be used to interpret the location part of users queries (see in Section IV-D).

2) Location Hierarchy: In our previous work [11], we analyzed the relationship among different locations in SYSLOG as a hierarchical structure. It can be used to model the generic location hierarchical structure in different data sets, shown in Figure 2. We classify the basic components in the location hierarchy into physical hierarchy and logical hierarchy. The components in the physical hierarchy have a clear hierarchical structure from top to bottom. The arrow in the figure illustrates a "one-to-many" relationship. For example, one router has multiple slots, each slot can have multiple ports and etc. Besides physical hierarchy, there are some logical components which can be mapped to some physical components. For example, one multilink/bundlelink can be mapped to multiple physical interfaces. After converting the logical components to physical components, we can construct a simple, clean *tree* structure of location hierarchy. Note that such hierarchy structure is *vendor independent*. The hierarchy describes the relationship between different locations. The indexing and searching algorithms in NetSearch are designed based on this structure.

B. Location Extraction

Extracting the detailed location information from the network message, such as SYSLOG message, is far from a trivial task. A brute force way is to utilize the message syntax which is provided in the vendor manuals. However, the message syntax keeps changing with the development of router software and hardware implementations, and it is also quite different across vendors. These complications motivate us to derive a generic method to extract the locations information from a network message with minimum requirement of knowledge on the message syntax. Due to the complexity of message syntax and structure, we use a hybrid approach by combining of the following three methods that are based on *router configuration*, *message context*, and *domain knowledge*. We will first explain these three methods in details separately, and then explain how to combine them together.

1) Based on Router Configuration: Recall in Section IV-A1 that we build location dictionary based on router configuration files. The location dictionary consists of pairs of location type and its possible values for each network element. To extract location information from a network message, we match each word in the message with the possible location values in the dictionary. If a match is found, we extract it from the message and consider it as the location of the message.

The advantage of this method is the high accuracy. Since the configuration file is a reliable data source, the extracted location dictionary should always be the accurate[2]. However, not all locations in the network messages are expressed in the same form as they are in the router configuration file. For example, a network message may contain "slot 2", while the simple value "2" is apparently not in configuration dictionary. Therefore, we need to find other complementary methods to extract more location information.

2) Based on Message Context: We observe that leading words following with the concrete value, in many cases, indicate the type of location in a network message. For example, leading words "Interface" and "slot" in messages "*Interface Multilink1034* change state to down" and "*slot 2* failure" give the indication on the location types and their values of the network messages. In order to extract location information based on message context, we pre-define several words as leading words that indicate the appearance of a location, and then match the context in each message with these leading words. This is a simple and lightweight method, without the help of external data sources. In particular, it is very useful to identify the locations like "port 2", which cannot be found using aforementioned location dictionary based method or later domain knowledge based method. The disadvantage of this method is that it may introduce some errors and the selection of leading words relies on certain domain knowledge.

3) Based on Domain Knowledge: The aforementioned two methods cannot guarantee to solve the location extraction problem completely. For example, the location "1/3" in message "the failure happens on [1/3]" is not able to be extracted by either of these two methods. Because it is a *variation* of location presentation, which cannot be found using configuration-based location dictionary. It cannot be identified by context-based method either since there is lack of leading word in the message. To extract such kind of locations, we need to know the location patterns in advance and hard-code these patterns into location extraction procedure.

4) Our Hybrid Approach: The three methods exhibit different level of dependency on the network domain knowledge with the configuration based method being the one relying least on the domain knowledge. Moreover, the configuration based method is the most reliable one among all three methods. Given these two observations, in practice, we first use the configuration based method to extract location. If it fails, then we apply the context-based method and then domain knowledge based method.

[2]This is certainly under the assumption that no location name is ambiguous. During our evaluation, we didn't find such cases based on our network data.

C. Indexing and Searching

After extracting the location information from messages, we index the network messages to serve the searching purpose. To design an indexing and searching system, we first need to consider the query requirements. The users of NetSearch are always interested in network events, that are annotated with time and space information. Moreover, the users of NetSearch may need to describe the events with other important information (e.g. link down). Therefore, NetSearch indexes messages based on three aspects of information: time, location, and other descriptions (optional). Given a query, the NetSearch searches messages on these three aspects in parallel. The messages that satisfy conditions on *all* three aspects are returned as the final results of the query. Temporal query and other description query are relatively simple. We can use existing indexing and searching tools like Zettair [12] to implement. In the rest of this part, we explain location query in detail.

Upon querying on location, NetSearch finds all relevant messages to the given location. As discussed in Section IV-A2, network messages location information naturally forms a tree structure after mapping the logical locations to physical locations. Therefore, we define that two locations are *relevant* if and only if their positions in the location hierarchy tree are either the same or one is the other's ancestor. Accordingly, the indexing process of location information is to construct the location hierarchy tree, and the searching process is to find all relevant nodes that satisfy the given the location expression in the query.

We use a concrete example (shown in Figure 3) to demonstrate the indexing and searching process. Initially, we have a number of messages with message IDs (e.g., 1 - 5) and location expressions (e.g., 0:1:0) on a router. First, we consider the router ID as the root of the tree. In order to tag the messages to proper positions of the tree, we will convert the expression of location into separated tokens. In this example, the interface "0/1/0" will be tokenized into three levels 0, 1, and 0, respectively. Note that different vendors may have different format of interface name, e.g., 0/1/0, or 0:1:0. But the structure of the hierarchy tree should be the same and the difference only lies in the delimiters in the location names. The combination of three level tokens (0,1,0) describes the hierarchical path from the root (i.e., the router) to a destination node. We start from the root node and follow this path to reach the destination node [3]. Then the message ID is associated with that destination node. Figure 3 shows the example after we tag five messages into the tree. Note that such an indexing process is continuous with the hierarchy tree being updated and new messages being tagged.

To search all relevant messages for a given location, we convert the query location to hierarchy path expression. If we can find the node by following the hierarchy path, then relevant messages are the *union* of messages that associated with destination node and its descendants. For example, upon searching for location "0:1", we find the node where message 4 is tagged as the destination node and return messages 4 and messages 1 and 5 (i.e., the messages that associate with descendant nodes of the destination node).

[3]A new node will be created if it does not exist during this process.

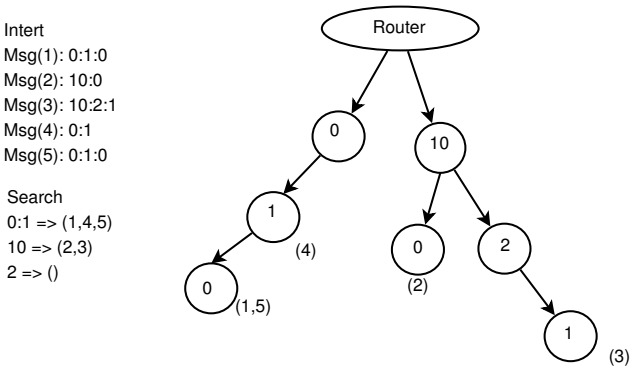

Fig. 3. The example to search locations

Furthermore, in some cases the queried location does not specify each level of the hierarchy. For example, the location "port 0" actually does not specify the location on the first level of hierarchy. Therefore, to search the query related to "port 0", we will search all nodes on the first level, then identify the second level with label 0.

D. Query Interpretation

Upon receiving a query, NetSearch needs to interpret the query. Recall that a typical query has three parts: time window, location, and optional description. The format of time window is well-defined. The optional description is treated as plain text. The subtle part is the location query.

NetSearch provides certain kind of flexibility of location query. Users have two choices. First, users can specify the type of location followed by its value, e.g., "Interface XX", "Circuit ID XXX". Second, if users do not specify the location type, NetSearch can automatically infer the location type based on the syntax signatures of the location value in the query.

The location type inference is based on two assumptions. First, different types of locations have different signatures. Second, these locations can be found in the location dictionary that we learned based on configuration files in Section IV-A1. Given these two assumptions, the problem is reduced to find the entries in the location dictionary whose values are identical to the queried location. However, brute force searching on the dictionary is time consuming because the dictionary is quite large (e.g., millions of entires). In order to speed up the searching process, we extract the signatures of the location dictionary. That is, instead of matching the exact value of location, we match the signature of each location.

More specifically, we find that signature of a typical location string can be expressed with the combination of three categories of characters: digits (D), alphabetic characters (A), and others (O). This observation is true for different router vendors. The reason is that the vendors usually use digits to denote different elements on the same location level (e.g. slot 1, slot 2), and use alphabets to name the location (e.g. Interfaces types like Serial, Ethernet), and use the other special characters to separate different location levels (e.g. five levels for SERIAL2/0.7/11:0). For this reason, we only record the categories of each characters as the signature within the location string. For example, interface "SERIAL2/0.7/11:0" can be expressed by "AAAAAADODODODDOD".

TABLE IV. LOCATION EXTRACTION EFFECTIVENESS ON SYSLOG MESSAGES

Data	Coverage				Error			
	Config	Context	Knowledge	Hybrid	Config	Context	Knowledge	Hybrid
SYSLOG	29.5%	54.6%	43.5%	100%	0%	3.6%	1.3%	1.4%

V. EVALUATION

We use three network measurement data sets (SYSLOG, OSPF and SONET) to evaluate three key components of NetSearch – location extraction, indexing and searching, and query interpretation. All these data sets are collected from a tier-1 ISP backbone network during one entire recent month. The network has around a couple of thousands of routers of multiple vendors. The routers in the network record millions of SYSLOG messages per day. There are tens of thousands of SONET and OSPF messages per day as well. Because routers are of multiple vendors, the formats of SYSLOG messages differ significantly across router vendors. NetSearch is designed to be vendor independent and handles message format variation well.

A. Location Extraction

In order to evaluate the effectiveness of our methods, we define two metrics as follows. The *coverage* is defined as the percentage of messages whose detailed location can be extracted by our methods. The *error* is defined as the percentage of erroneous extractions among all extracted locations.

Table IV shows the effectiveness of location extraction of router SYSLOG messages, when applying three different methods separately and simultaneously. Since detailed locations from SONET and OSPF messages are already presented in one of message fields, there is no need to extract. Only SYSLOG data is evaluated in this table. We have several observations on the results. First, the error value of config-based method is zero. It indicates that config-based method is the most reliable method. It is also the reason we choose it as the very first method in the hybrid approach by combining three methods. Second, the coverage of each method is limited (all less than 60%). It implies that no single method can extract location effectively. Third, the hybrid approach of combining all three methods can cover almost all messages, and only yields 1.4% of errors.

B. Indexing and Searching (of Locations)

We focus on evaluating the performance of location indexing and searching here, since it is the critical and novel part of NetSearch. We evaluate the response time of location query of the *combination* of three data sets. We randomly select a number of queries across different levels of hierarchy tree. By default, the time window of the query is fixed as one day. Figure 4 shows the average response time (in milliseconds) with confidence intervals when varying the number of queries. We find that the total response time increase linearly when the number of queries increases. On average, it only takes several seconds to respond to one thousand queries. In other words, the response time for a single query is on the order of several milliseconds. More detailed evaluation of indexing and searching can be found in our technical report [13].

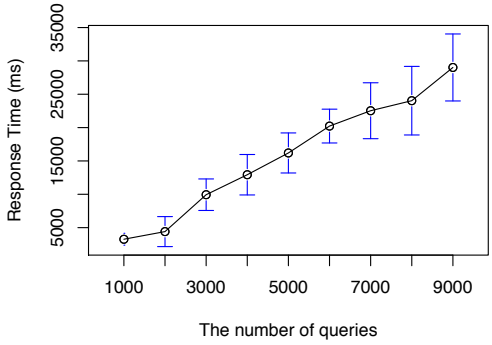

Fig. 4. The response time of location query

TABLE V. THE EFFECTIVENESS OF SIGNATURE EXTRACTION

Type	# values	# Signatures	Ratio
Interface	127431	234	1.836e-03
VRF	15379	31	2.016e-03
CIRCUIT	385550	8	2.074e-05

C. Query Interpretation

In Section IV-D, we explained how NetSearch interpret users' location query by extracting the signatures of location within the location dictionary (Note that it is independent from data sets). Table V shows the number of signatures and the number of locations extracted based on router configurations of one day. The ratio in this table is defined as the number of signatures divided by the number of location values. Here we demonstrate three location types: interface, VRF [4] and CIRCUIT [5]. We find that signature extraction can greatly reduce the size of the location dictionary (from 3 to 5 orders of magnitude). The signatures of location dictionary can be as small as several megabytes, and hence can be loaded in the memory to achieve online query interpretation.

VI. APPLICATIONS

In this section, we use one example to demonstrate that NetSearch can significantly benefit the network operation tasks. More applications of NetSearch can be found in our technical report [13]. One primary application of NetSearch is to assist impact analysis of known network events. For instance, network operators may want to know the network wide impact of a link failure or configuration change. Using NetSearch, we can query by the time window and location of the event to obtain all related messages across multiple relevant routers, and analyze the impact based on these messages.

[4]VRF stands for Virtual Routing and Forwarding. It is a technology that allows multiple instances of a routing table to co-exist within the same router at the same time. VRF is a common technique used in VPN environment. The VRF ID XXX:XXXX is a simple a conceptional name.

[5]A circuit ID is a company-specific identifier assigned to a data or voice network between two locations.

TABLE VI. SYSLOG MESSAGES BY GREPPING ROUTER R1 AND PORT 1/1/1

Date	Time	Message type	Detailed messages
2011-03-02	10:02:26	SNMP_TRAP_LINK_DOWN	ifIndex 296, ifAdminStatus up(1), ifOperStatus down(2), ifName 1/1/1
2011-03-02	10:02:27	SNMP_TRAP_LINK_UP	ifIndex 296, ifAdminStatus up(1), ifOperStatus up(1), ifName 1/1/1 dv

TABLE VII. SONET MESSAGES BY GREPPING ROUTER R1 AND PORT 1/1/1

Timestamp	Router	Interface	Circuit ID
2013-03-02 04:00:00	R1	T1_1/1/1:27.0	DHEC 12345.802
2013-03-02 06:00:00	R1	T1_1/1/1:27.0	DHEC 12345.802
2013-03-02 10:00:00	R1	T1_1/1/1:27.0	DHEC 12345.802

TABLE VIII. INTERFACES AND IP ADDRESS MAPPING. "–" MEANS NO VALID IP ADDRESS ASSIGNED.

Router	Interface	IP
R1	T1_1/1/1:27.0	192.168.1.9
R1	T1_1/1/1:2	–
R1	T1_1/1/1:1.0	192.168.1.33
R1	T1_1/1/1:1	–
R1	T1_1/1/1:14	–
R1	T1_1/1/1:2.0	192.168.1.81
R1	T1_1/1/1:14.0	192.168.1.125
R1	T1_1/1/1:27	–
R1	CT3_1/1/1	–

TABLE IX. INTERFACE AND CIRCUIT ID MAPPING

Router	Interface	Circuit ID
R1	T1_1/1/1:2	DHEC 12345.802
R1	T1_1/1/1:1	DHEC 12345.801
R1	T1_1/1/1:14	DHEC 1234.802
R1	T1_1/1/1:27	DHEC 123456

In a concrete example, the network operator of the tier-1 ISP noticed that the Port 1/1/1 on router R1 was unstable during one recent week. To analyze the impact of such an event without using NetSearch, one can grep the router R1 and port 1/1/1 on measurement data sets. This is usually the first thing that the network operator does in performing an impact analysis. Note that such brute force searching is extremely time consuming given the sheer message volume within the relative large time window. It typically takes roughly ten minutes for SONET data and half an hour for SYSLOG data. Tables VI and VII show the examples of messages from SYSLOG and SONET data, respectively. The messages from SONET data and those from SYSLOG data do not always occur closely on time. This indicates that they may not be related to each other. Moreover, these message do not provide a complete picture of the impact of unstable port 1/1/1 on router R1. To find more related messages, we need to know all the locations that are relevant to port 1/1/1. By extracting implicit location mapping information manually from the configuration file, we find that there are nine interfaces related to the port 1/1/1, including a Channelized T3 (CT3) interfaces on the high level, and eight T1 interfaces. Moreover, we also find that several IP addresses assigned to these interfaces. The mappings among router, interfaces and IPs are shown in Table VIII. For some interfaces, we also find their corresponding circuit ID (shown in Table IX), which may appear in some related physical layer messages.

Given location information, we can further grep the related interfaces, IP addresses and circuit IDs to find more relevant messages. More specifically, IP addresses can be used to search in OSPF data sets; Circuit ID can be used to search in SONET data; all location information can be used to search in SYSLOG. After the extensive searching, we observe not only link flap but also BGP session failures in SYSLOG, which requires a sequence of location mappings that are extremely challenging to do manually in a short time: from *BGP neighbor IP* to a *local interface IP* then to *interface name*. We can as well find some related OSPF monitoring messages given the corresponding IP addresses. In total, there are over 200 alarm messages from SONET data, about 100 link flap messages and 60 BGP flap messages from SYSLOG data and 100 messages from OSPF data.

To briefly summarize our observation of the impact of this outage event:

- The port is out of order multiple times within the investigation window.
- Some outages on layer-1 are very short (observed from SONET), and therefore, have a negligible impact on the higher layers (no corresponding observations in SYSLOG).
- Some other layer-1 outages are relatively long. It can cause the link flap, OSPF updates (observed in OSPF) and BGP session failures (and thus have an impact on customers).

So far, we describe the manual way to extract all relevant messages for a given event of interest. As we mentioned, it is a time consuming and complicated task. First, the simple grep cannot get the complete big picture on the event impact. Second, the process heavily relies on domain knowledge on network configuration in order to uncover the implicit relationship and mapping among different locations and network protocols. Finally, the process may take a long time, e.g., several hours. In contrast, we can input a single query into NetSearch and obtain the same set of messages within one minute. Therefore, NetSearch significantly speeds up and simplifies the network diagnosis process.

VII. RELATED WORK

Some toolsets have been developed to provide sophisticated query and data management facilities for network management purposes [14], [15], [16]. However, these tools fail to consider the spatial relationship of different data entries and therefore cannot be directly used for comprehensive searching purpose.

Exploring spatial patterns is extremely important in database and network areas. Database researchers proposed a series of data structures and algorithms for efficient spatial searching [17], [18], [19]. However, they focus on the geographic or geometric locations, which is different from the spatial model of network events. Recently, networking researchers pay more attenion to the temporal and spacial patterns in

term of network event descriptions. For example, Wang [20] proposed a general framework of learning, indexing, and identifying topological and temporal correlation existing in network event data. However, topological information they used is limited on router-level (e.g. neighborhoods of routers). Our work presents a generic spatial description of network event locations. In our previous work [11], we developed a system to automatically group relevant SYSLOG messages by exploring the temporal and spatial property in SYSLOG messages. In comparison, our study in this paper focuses on searching rather than grouping. Moreover, the methodology is applied to different types of network messages, rather than restricted to SYSLOGs.

VIII. Conclusion

In this paper, we develop a system called NetSearch that can "google" a wide range of network data. NetSearch pre-processes various network data sources including device logs, alarms and outputs from network measurement systems, constructs indices that match an abstract network spatial hierarchy model and the inherent timing information as well as the textual information contained in the data, and efficiently retrieves and effectively presents the relevant information that network operators search for. We evaluated NetSearch using real-world network data collected from a large tier-1 ISP network and demonstrated how NetSearch can be applied on complex network management tasks such as impact analysis. We believe NetSearch will become an essential building block for many network management tools. Applying NetSearch on more data sets, integrating it into various network management tools, and using Hadoop to further improve search response time are among our future work.

References

[1] C. R. Kalmanek, Z. Ge, S. Lee, C. Lund, D. Pei, J. Seidel, and J. Y. Jacobus van der Merwe, "Darkstar: Using exploratory data mining to raise the bar on network reliability and performance," in *Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop on*. IEEE, 2009, pp. 1–10.

[2] "The data-driven approach to network management: Innovation delivered," http://www.research.att.com/articles/featured_stories/2010_05/201005_networkmain2_article.html.

[3] "Emc lonix website," http://www.emc.com/products/family/ionix-family.htm.

[4] "IBM netcool website," http://www-01.ibm.com/software/tivoli/welcome/netcool.

[5] B. Yuwono and D. Lee, "Search and ranking algorithms for locating resources on the World Wide Web," in *ICDE*. Published by the IEEE Computer Society, 1996, p. 164.

[6] S. Rice and S. Bailey, "Searching for sounds: A demonstration of FindSounds. com and FindSounds Palette," in *Proc. of the International Computer Music Conference*. Citeseer, 2004, pp. 215–218.

[7] J. Lee, "Software learns to tag photos," *Technology Review*, vol. 9, 2006.

[8] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 4, pp. 749–762, 2008.

[9] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, ser. IMW '02, 2002, pp. 197–202.

[10] A. Shaikh and A. Greenberg, "OSPF monitoring: architecture, design and deployment experience," in *Proceedings of NSDI*. Berkeley, CA, USA: USENIX Association, 2004, pp. 5–5. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251175.1251180

[11] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, "What happened in my network: mining network events from router syslogs," in *Proceedings of ACM IMC*. ACM, 2010, pp. 472–484.

[12] J. Zobel, H. Williams, F. Scholer, J. Yiannis, S. Heinz, N. Lester, W. Webber, A. Moffat, and A. Vo, "The zettair search engine," *Search Engine Group, RMIT University, Melbourne, Australia*, 2004.

[13] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, "Netsearch: Googling large-scale network management data," http://www.cc.gatech.edu/~tongqqiu/publication/netSearch_tech.pdf, Tech. Rep., 2011.

[14] G. Fowler and B. Krishnamurthy, "dss – data stream and scan," AT&T Lab – Research, Tech. Rep., 2005.

[15] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk, "Gigascope: A stream database for network applications," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003, pp. 647–651.

[16] M. Feridun and A. Tanner, "A search engine for systems management," in *Proceedings of the 12th IEEE/IFIP Network Operations and Management Symp*, 2010, pp. 697–210.

[17] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '84, 1984, pp. 47–57.

[18] J. Orenstein, "Spatial query processing in an object-oriented database system," in *Proceedings of the 1986 ACM SIGMOD international conference on Management of data*. ACM, 1986, pp. 326–336.

[19] H. Six and P. Widmayer, "Spatial searching in geometric databases," in *Data Engineering, 1988. Proceedings. Fourth International Conference on*. IEEE, 1987, pp. 496–503.

[20] T. Wang, M. Srivatsa, D. Agrawal, and L. Liu, "Spatio-temporal patterns in network events," in *Proceedings of ACM CoNext*, 2010, pp. 3:1–3:12. [Online]. Available: http://doi.acm.org/10.1145/1921168.1921172