Visualization of Network Data Provenance

Preprint Version
Forthcoming in Workshop on Massive Data Analytics on Scalable Systems (DataMASS 2012),
co-located with High Performance Computing Conference (HiPC)

Peng Chen*, Beth Plale*, You-Wei Cheah*
Devarshi Ghoshal*, Scott Jensen* and Yuan Luo*
*School of Informatics and Computing
Indiana University, Bloomington, IN, USA
{chenpeng,plale,yocheah,dghoshal,scjensen,yuanluo}@indiana.edu

***Abstract*—Visualization facilitates the understanding of scientific data both through exploration and explanation of the visualized data. Provenance also contributes to the understanding of data by containing the contributing factors behind a result. The visualization of provenance, although supported in existing workflow management systems, generally focuses on small (medium) sized provenance data, lacking techniques to deal with big data with high complexity. This paper discusses visualization techniques developed for exploration and explanation of provenance, including layout algorithm, visual style, graph abstraction techniques, and graph matching algorithm, to deal with the high complexity. We demonstrate through application to two extensively analyzed case studies that involved provenance capture and use over three year projects, the first involving provenance of a satellite imagery ingest processing pipeline and the other of provenance in a large-scale computer network testbed.**

I. Introduction

Distributed applications that are run on distributed frameworks such as network testbeds and clouds, can have hundreds to thousands of communicating nodes and messages, and understanding this complex behavior is crucial. Visualizing the data from a network experiment, including its structure and measurement, is a technique used to facilitate this understanding. However, visualization of network data alone may not be enough information for researchers to fully understand and accept the results, because missing are data sources and data processing services used to derive the data and which intermediate data were produced during the derivation process. In other words, a researcher cannot know the reason behind network behavior unless they see a picture of the complete and complex cause and effect. In fact, a researcher may need to visually see the combination of network data and its provenance, which together gives the meta-data about network data and how the data are generated.

Data provenance, specifically, is the lineage of a data product or collection of data [17]. More broadly, it can describe data that is observational or imagery data arriving in real time from sensors, networks of sensors, and instruments; data from large-scale distributed applications; results from computational models and data mining; field studies such as documenting human use of a plot of land over time; regional positional data; scholarly reports in journals, etc. Provenance can identify event causality; enable broader forms of sharing, reuse, and long-term preservation of scientific data; can be used to attribute ownership and determine the quality of a particular data set [18].

Iliinsky and Steele (2011) [10] identify two categories of data visualization: exploration and explanation. *Exploratory visualization* is visualization designed to support a researcher who has large volumes of data and not certain what is in it. *Explanatory visualization*, on the other hand, is visualization that takes place when a researcher knows what the data has to say, and is trying to tell that story to someone else. The two serve different purposes, and there are tools and approaches that may be appropriate for one and not the other.

Visualization of data provenance that has network characteristics is more sophisticated than visualizing pure network data. Network data can be conveyed through visualization of bandwidth, overload, and Round-trip time (RTT) time, whereas network data provenance visualization should include capabilities for visualizing messages/packets, the data transfer process, and any related metadata. Network data provenance visualization differs also from prior provenance visualizations in that the network data provenance can be large, necessitating techniques such as navigation, abstraction, and layout algorithms to make the large graph meaningful.

In this paper, we introduce a number of provenance visualization techniques for interactive navigation, manipulation, and analysis of large-scale data provenance. We demonstrate our visualization tool suite for both exploratory and explanatory types of visualization. In fact, from our experience both are needed; where exploratory visualization is useful at the data analysis phase and explanatory visualizations at the presentation phase. We provide evidence of the visualization techniques applied to two case studies that we engaged in over a three year period. While common capabilities such as viewing attributes, manipulating, and comparing graphs can be shared, visualizations have different content and user interests, so customized layout algorithm and visual style are also needed. The visualization techniques are implemented into a Cytoscape [16] plugin.

The remainder of the paper is organized as follows. Section II gives related work. Section III gives an overview of the applications from which provenance is captured. Section IV identifies and describes the multiple provenance visualization techniques we develop. Sections V and VI give evidence of application of the techniques. Finally we conclude in Section VII with a discussion of future work.

II. Related Work

Kunde et. al [12] derive abstract types of user requirements for provenance visualization, including: 1) process: the sequence of the process steps is in the center of inspection; 2) results: the intermediate or end results of interactions are in the center of the users view; 3) relationship: the relationship of interactions or actors is important; 4) timeline: the time is important to observe; 5) participation: the correctness of the participants is important; 6) compare: the comparison of two subjects shows the difference between them; 7) interpretation: an individual visualization view depending on the special question of the end-user.

The goal of our visualization research is to serve both broad and narrowly focused audiences, so addresses each of the above requirements as follows: 1-3) our visualization tool is based on an accepted model for provenance representation provenance, namely, Open Provenance Model (OPM) [13], which denotes entities (processes and artifacts) as nodes, and relationship as edges in a graph. It is able to show a complete graph with both process steps and intermediate (final) results, or abstract graphs focusing on either one of them; 4) OPM is capable of representing time information to nodes and edges, and our visualization tool has a special function called Play movie by time that can help the user understand the dynamics through time; 5) participation is represented by agents through "was controlled by" relationship in OPM, so our tool helps the user visually evaluate the correctness of participations; 6) users can compare attributes of nodes using our tool. We also improve the graph matching algorithm introduced by DePiero [6], and use it to compare two provenance graphs; 7) for the last type of user requirement (Interpretation), we will show how we satisfy it with customized layout algorithm and visual style in our use cases.

Prior provenance visualization tends to deal with small graphs, and seldom addresses the issue of graph layout and graph matching. Specifically, Taverna [15] uses visualization to help answer questions that establish how the experiment results were obtained; VisTrails [9] allows users to navigate workflow versions in an intuitive way, to visually compare different workflows and their results, and to examine the actions that led to a result; Probe It! [4] enables scientists to move the visualization focus form intermediate and final results to provenance back and forth; The Prototype Lineage Server [1] allows users to browse lineage information by navigating through the sets of metadata that provide useful details about the data products and transformations in a workflow invocation; Pedigree Graph [14], one of tools in Multi-Scale Chemistry (MSC) portal from the Collaboratory for Multi-Scale Chemical Science (CMCS), is designed to enable users to view multi-scale data provenance; The MyGrid project renders graph-based views of RDF-coded provenances using Haystack [20]; Provenance Explorer [2], a secure provenance visualization tool,dynamically generates customized views of scientific data provenance that depend on the viewer requirements and/or access privileges.

Provenance Map Orbiter [13] uses graph summarization and semantic zoom to navigate large provenance graphs. It gives a high-level abstracted view of a graph and the ability to incrementally drill down to the details. However, node summarization depends on having available sufficient semantic information. Too, a summarized view may not work well for visual comparison of multiple graph components at a detailed level.

The visualization tool we developed can satisfy different types of user requirements, and can handle large-scale visualization with customized layout algorithm and visual styles.

III. Scenarios of Use

The provenance used in this study is drawn from two projects in which we engaged over a 3 year period. The first project examined the utility of provenance in computer network applications. These applications are often large scale distributed applications that run on network simulators, or on large testbed networks such as PlanetLab, ORBIT, and other platforms that are part of Global Environment for Network Innovations (GENI) [8]. GENI provides collaborative and exploratory networking environments for research - at multiple layers in the networking stack, including slice creation, topology of the slice, operational status, and links to measurement data. Our challenge in this project is to capture provenance from the layers of the network stack without affecting or instrumenting the application. The question we were asking is whether enough meaningful provenance can be captured at the network stack level, and whether provenance adds to the understanding of the user. One particular experiment we support studies denial of service (DoS) attacks in a WiMAX network [5]. Researchersanalyze the impact that improper configuration of one or more subscriber stations (a WiMAX network is made up of one base station that serves one or more subscriber stations) increases or decrease DoS vulnerabilities of WiMAX networks.

The second project involves capture of provenance of a NASA satellite imagery ingest processing pipeline. The instrument generating the images is the AMSR-E (Advanced Microwave Scanning Radiometer, Earth Observing System). It sits aboard a polar orbiting satellite and produces products about the poles. We initially focus on the process that produces imagery of sea ice. The question answered in the project is whether provenance can provide a more useful form of lineage information than is captured by the existing pipeline processing, which is located at the University of Alabama Huntsville. The limitations of the existing information capture in the processing pipeline are several:

- Full lineage information is not collected.
- Sometimes complete input information is not collected.
- There is no mechanism for the user to request information on previous or more recent versions of a given product.

- Provenance information is embedded in the data. Existing provenance-like information is captured in inventory metadata and stored in the files.
- Comparison of two versions of data products is cumbersome. To compare two different versions of a data product, a data user must first find the release notes for each version and then use a tool like HDFView to analyze the limited provenance metadata stored in the files.

Both projects use the Karma provenance capture tool [19], a standalone system that can be added to existing cyberinfrastructure for purposes of collection and representation of provenance data.

IV. Provenance Visualization Techniques

The goal of provenance visualization is to help a user navigate provenance. A researcher brings a mental map of what is going on in an experiment, and uses this model to interact and explore the provenance. We develop a number of visualization techniques that we discuss here. The techniques are implemented into a plugin to Cytoscape, an open source software platform for complex network analysis and visualization. Cytoscape is appropriate for provenance visualization because of its support for detail and overlaying visualizations with additional annotations. The Cytoscape plugin can generate the provenance graph visualization by interacting with the Karma provenance server to extract provenance in the form of a graph as XML.

A. Incremental Loading

Provenance can be very large, not only because a lineage record can be long but because OPM v1.1 [13] supports key value annotations to provenance graphs, the latter of which opens the opportunity for capture of extended information about execution or object creation. To better support visualizations over large graphs, we support reads of provenance graph in XML format with and without annotations. Annotations can be separately queried through the Karma query API. That is, the Karma system generates an OPM compliant XML file that does not have annotations to process or artifacts, and if the visualization tool loads this XML file, it will also establish a background connection to the Karma server, to retrieve annotations when some process or artifact is selected during navigation. This incremental loading allows loading of annotations on demand.

During the interactive visualization, the user may change the graph and the node (edge) attribute, so the visualization tool also supports saving provenance data from Cytoscape to an OPM compliant XML file.

B. Customized Layout

Layout is a key element to increasing researchers' understanding of large-scale network data provenance. Layout depends on both the nature of provenance data and the user requirement. In fact, it often takes seeing multiple layouts for a researcher to judge which is most meaningful. Cytoscape offers several default layouts that are meaningful for provenance visualization. For example, its hierarchical layout organizes a provenance graph into layers based on relationship such that first causes appear at the top of a visualization and final effects appear at the bottom.

In addition, we designed several customized layouts including 1) extended hierarchical layout algorithm that sorts sibling nodes by their time order, 2) group of layout algorithms specifically for computer network provenance, and 3) string-embed based layout algorithm for provenance data like a history chain. The last-named is illustrated in Figure 1 where provenance is shown for 1 month processing of a NASA satellite ingest processing pipeline. The magenta nodes are data products and green nodes are process of the workflow, and each cluster corresponds to the provenance of a daily processing. At the left-end is a cluster of workflows; this is the result of creating a month-long sea-ice data products. The provenance graphs are connected because in creating a sea ice product one day, a moving window mask of prior days is used to locate the boundary between land ice and sea ice.

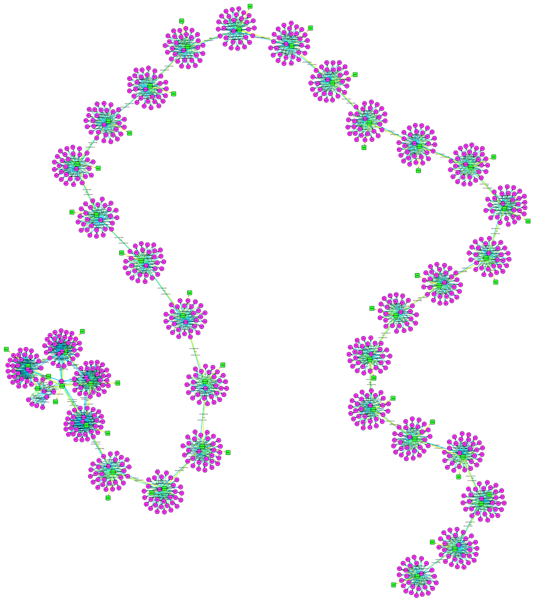


Fig. 1. Provenance visualized in a history chain shows the relationship between daily products (each clover flower in the clover leaf chain) and final monthly products at the left-end.

C. Visual Style

One of Cytoscape's strengths is its ability to allow a user to encode any attribute of data (e.g., name, type, degree, weight, expression data) as a visual property (such as color, size, transparency, or font type). A set of these encoded or mapped attributes is called a Visual Style. We create a default visual style for provenance graphs, using green to color processes, magenta for artifacts, red for agents, and different colors and arrow styles for different types of edges. We also develop advanced visual styles to expose information from attributes.

For example, in the provenance we gathered from a WiMAX denial of service experiment (see case study), we developed a visual style which we call “WiMAX DoS Vis” that uses color coding to discriminate between the normal nodes and the attacker nodes; and between packets dropped and packets successfully received.

Additionally, using customized node graphics, a multi-line chart in this case, we show the statistics on traffic packets (Figure 2).

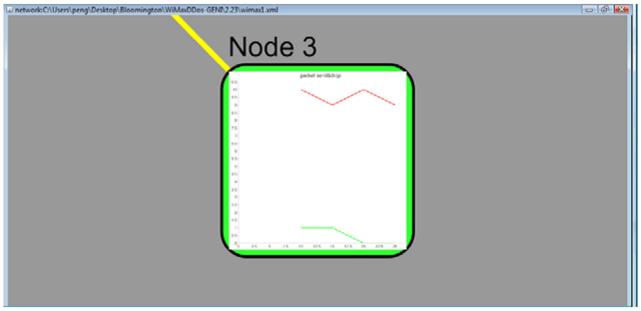

network:C:\Users\peng\Desktop\Bloomington\WiMaxDDos-GENI\2.23\wimax1.xml
Node 3
packet send&drop

Fig. 2. Using a special visual style developed using customized node graphics, a user can view statistics of packets that have been dropped and sent. The blue plot at bottom gives number of packets dropped, and the red plot represents the number of packets sent, both sampled every 5 seconds.

D. Abstract Views

The complexity of the provenance relationships can result in graphs that overwhelm the researcher. We identify two approaches to abstracting out complexity, including:

1) clustering neighbor nodes; and

2) process/artifact elimination.

Neighbor nodes can be clustered and reduced to a single node by an action in our Cytoscape plugin. This is useful when exploring a provenance graph and could be applied to deal with graphs having a large number of artifact nodes.

OPM v1.1 introduces two causal dependencies “was triggered by” and “was derived from” as summary edges for a process view (where an intermediary artifact was unknown) and a data view (where an intermediary process was unknown), respectively. It also provides completion rules that can be used to transform a graph between a complete view, a process view, and a data view. We developed two techniques of transformation based on completion rules, namely, process elimination and artifact elimination.

Process elimination eliminates all process nodes that link two artifact nodes with an outgoing edge “used” and an incoming edge “was generated by”. The process node is replaced by a new edge “was derived from”. Figure 3 shows a complete provenance view, and Figure 4 shows its data view after applying process elimination.

Artifact elimination eliminates an artifact node in the case where the outgoing edge of an artifact represents the relationship “was generated by” and its incoming edge represents the relationship “used”. The artifact will be replaced by a new edge between these two process nodes that represents the relationship “was triggered by” (Figure 5).

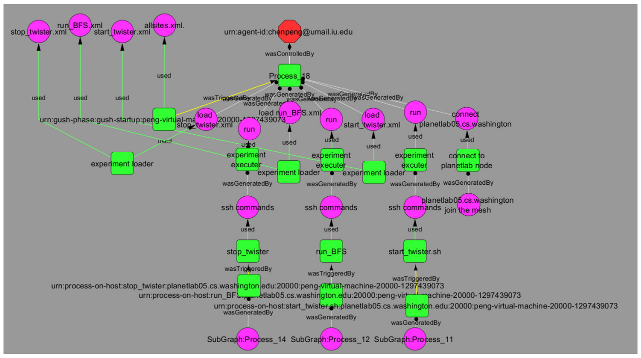

stop_twister.xml
run_BFS.xml
start_twister.xml
allsites.xml.
urn:agent-id:chenpeng@umail.iu.edu
wasControlledBy
Process_18
used
wasTriggeredBy
wasGeneratedBy
urn:gush-phase:gush-startup:peng-virtual-machine-20000-1297439073
load stop_twister.xml
load run_BFS.xml
run
load start_twister.xml
connect planetlab05.cs.washington
experiment loader
experiment executer
experiment excuter
connect to planetlab node
ssh commands
planetlab05.cs.washington join the mesh
stop_twister
run_BFS
start_twister.sh
urn:process-on-host:stop_twister:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
urn:process-on-host:run_BFS:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
urn:process-on-host:start_twister.sh:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
SubGraph:Process_14
SubGraph:Process_12
SubGraph:Process_11

Fig. 3. Complete provenance for a breadth-first search application

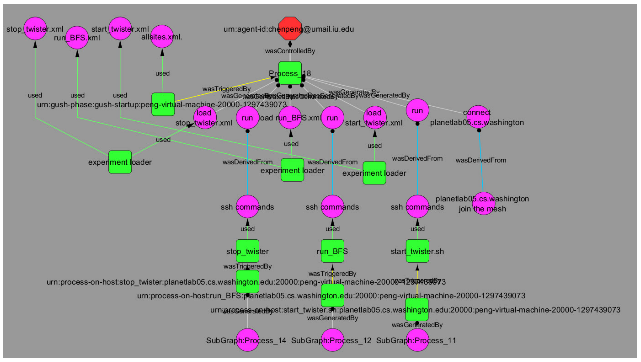

stop_twister.xml
run_BFS.xml
start_twister.xml
allsites.xml.
urn:agent-id:chenpeng@umail.iu.edu
wasControlledBy
Process_18
used
wasTriggeredBy
wasGeneratedBy
wasDerivedFrom
urn:gush-phase:gush-startup:peng-virtual-machine-20000-1297439073
load stop_twister.xml
load run_BFS.xml
run
load start_twister.xml
connect planetlab05.cs.washington
experiment loader
ssh commands
planetlab05.cs.washington join the mesh
stop_twister
run_BFS
start_twister.sh
urn:process-on-host:stop_twister:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
urn:process-on-host:run_BFS:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
urn:process-on-host:start_twister.sh:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
SubGraph:Process_14
SubGraph:Process_12
SubGraph:Process_11

Fig. 4. Breadth-first search application after process elimination

E. Graph Comparison

Comparing two provenance graphs is a key piece of provenance analysis. We improve and implement the Direct Classification of node Attendance (DCA) algorithm to compare two provenance graphs by finding the best matched subgraph and unmatched nodes.

Direct classification of node Attendance finds isomorphisms between graphs and subgraphs. The method first evaluates evidence describing the likelihood of a node’s predicted attendance in another graph. The evidence is based on measures that are local to each node, including node connectivity, node, and edge attributes. It then finds a node-to-node mapping

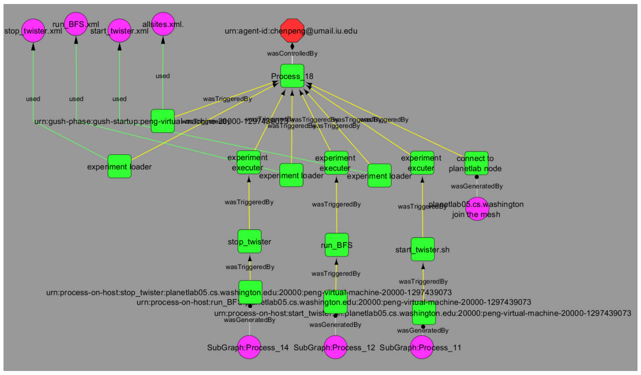

stop_twister.xml
run_BFS.xml
start_twister.xml
allsites.xml.
urn:agent-id:chenpeng@umail.iu.edu
wasControlledBy
Process_18
used
wasTriggeredBy
wasGeneratedBy
urn:gush-phase:gush-startup:peng-virtual-machine-20000-1297439073
experiment loader
experiment executer
experiment excuter
connect to planetlab node
planetlab05.cs.washington join the mesh
stop_twister
run_BFS
start_twister.sh
urn:process-on-host:stop_twister:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
urn:process-on-host:run_BFS:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
urn:process-on-host:start_twister.sh:planetlab05.cs.washington.edu:20000:peng-virtual-machine-20000-1297439073
SubGraph:Process_14
SubGraph:Process_12
SubGraph:Process_11

Fig. 5. Breadth-first search application after artifact elimination

and reorders nodes to permit a direct comparison to be made between the resultant graphs. The algorithm is of polynomial order. It yields approximate results, maintaining a performance level for subgraph isomorphisms at or above 95% under a wide variety of conditions and with varying levels of noise. The performance on the full size comparisons associated with graph isomorphisms has been found to be 100/100, also under a variety of conditions [6]. The result of original DCA algorithm depends on the input order. That is, the result of matching graph A to graph B is different from matching B to A. We improve the DCA algorithm by making the matching process consistent regardless of the input order. Besides, the original algorithm forms the matching subgraphs purely based on value of node attendance, and we improve that by considering the nodes topology while forming matching subgraphs.

Below is the improved version of the DCA algorithm.

```
    // Compare each pair of nodes in each graph
 1: for all nodes $n_i$ in $G_1$ do
 2:     for all nodes $n_j$ in $G_2$ do
            // Always find best matching edges and adjacent nodes for node with larger degree
 3:         if number of incident edges: $n_i < n_j$ then
 4:             $n_1$= $n_i$ and $n_2$ = $n_j$
 5:         else
 6:             $n_1$= $n_j$ and $n_2$ = $n_i$
 7:         end if
            // Compare edges incident to current nodes
 8:         for all edge $e^1_k$ incident to $n_1$ do
 9:             for all edge $e^2_l$ incident to $n_2$ do
10:                 Compare connectivity along $e^1_k$ with $e^2_l$
11:                 Compare edge properties of $e^1_k$ with $e^2_l$
12:                 Compare adjacent nodes' properties of $n_1$ with $n_2$
13:                 Combine results of step 10 to 11 via Independent Opinion Pole (IOP)
14:             end for
15:             Save comparison of best matching edges and adjacent nodes
16:         end for
            // Compare current nodes
17:         Compare node properties of $n_i$ with $n_j$
            // Find similarity of current nodes
18:         Combine result of step 15 and 17 to form attendance rating of $n_i$ to $n_j$, namely $attendance(n_i, n_j)$
19:     end for
20: end for
    // Form matched subgraph
21: Let list of matched node-pair: $S \leftarrow empty$
22: Let list of current matching node-pair: $P \leftarrow empty$
    // Find the node-pair with peak attendance $A_1$
23: Let peak attendance $A_1 \leftarrow 0$
24: Let best matching node-pair $P \leftarrow null$
25: for all nodes $n_i$ in $G_1$ do
26:     for all nodes $n_j$ in $G_2$ do
27:         if $A_1 < attendance(n_i, n_j)$ then
28:             $A_1 \leftarrow attendance(n_i, n_j)$
29:             Let $P \leftarrow < n_i, n_j >$
30:         end if
31:     end for
32: end for
    // Add the best matching node-pair into $L$
33: if $A_1 > THRESHOLD$ then
34:     add node-pair $P$ into $L$
35:     add node-pair $P$ into $S$
36: else
37:     return empty matched graph
38: end if
39: while L is non-empty do
40:     remove node-pair $< n_i, n_j >$ from $L$
        // find the node-pair $< n^i_k, n^j_l >$ with peak attendance $A_2$
41:     for all  node $n^i_k$ connected to $n_i$ do
42:         Let peak attendance $A_2 \leftarrow 0$
43:         for all  node $n^j_l$ connected to $n_j$ do
44:             if $A_2 < attendance(n^i_k, n^j_l)$  then
45:                 if node-pair $< n^i_k, n^j_l >$ has not been matched in $S$ before then
46:                     $A_2 \leftarrow attendance(n^i_k, n^j_l)$
47:                 end if
48:             end if
49:         end for
50:         if $A_2 > THRESHOLD$ then
51:             add $< n^i_k$ , $n^j_l >$ into $L$
52:             add $< n^i_k$ , $n^j_l >$ into $S$
53:         end if
54:     end for
55: end while
56: Return $S$
```

Step 3 to 6 is the change we made from the original DCA algorithm to make the result of step 18 independent of input order and more reasonable. Considering an extreme case in the original DCA algorithm, where node n_1 in step 8 has only one edge while node n_2 has many, then step 15 will be executed only once and the result of step 18 will depend on whether that only edge of n_1 could find a good match. Step 21 to 55 in our implementation is a greedy algorithm that takes the node attendance as well as its topology into consideration while forming subgraph.

An example of comparing provenance graphs of two workflows from the AMSR-E ingest pipeline processing is shown in Figure 6, where the nodes in cyan on the left cannot be matched to corresponding nodes in the comparison graph on the right.

V. Exploratory Data Visualization

Visualization can be for purposes of exploration or explanation [10]. *Exploration* visualization is generally best done at a high level of granularity [10]. In this section, we show 1) layout algorithms and abstraction techniques that can reveal exceptions, and 2) layout algorithm, visual style and interactive

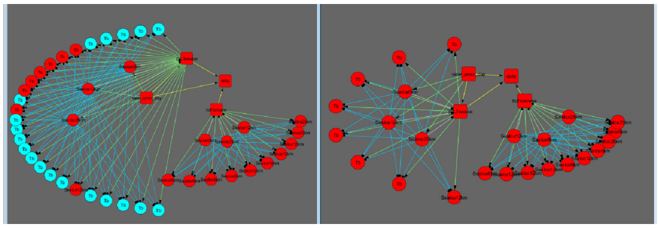


Fig. 6. Two workflows compared using improved DCA. Red nodes are matched to corresponding nodes in comparison graph. Left hand graph has 19 extra data products that cannot be paired with graph on right.

navigation can help determine why a WiMAX sender has higher throughput than others.

A. Breadth First Search Graph Coloring

The breadth first search experiment executes a graph coloring program using MapReduce [7] running on PlanetLab [3]. A head node partitions tasks to worker nodes; the experiment had one master node and ten slave nodes. However, the provenance visualization of this experiment has far too many artifact nodes connected to an extremely small number of process nodes. The reason for this is twofold: the application uses a head node to control worker nodes. Secondly, the philosophy we adopted in capturing provenance in applications that run in network frameworks such as GENI is that we would not capture provenance at the application level, but only at the middleware level. That is, we would not instrument the application itself. Provenance is captured from the software layers below the application. Since in this example, the MapReduce application and BFS were tightly coupled, both are treated as the application. Hence no provenance capture was done at the worker nodes. As a result, all resulting data products are tied to the couple processes that started the application. When the number of nodes of one type is orders of magnitude larger than the number of nodes of another type (processes), this presents problems for navigation and understanding.

Instead of manipulating the complete view, we utilize the clustering neighbor nodes functionality discussed in Section IV-D to create an abstract view. Figure 7 shows the abstract view under the hierarchical layout. By clustering data artifacts, we are able to see where and how each process ran, and what output (subgraph) was generated. A deeper investigation of the results shown in Figure 7 reveals that the last (rightmost) BFS process was not triggered successfully, and the application as a whole failed without output.

B. Computer Network Denial of Service

We experimented with application of our visualization approaches for exploratory visualization on a computer networking denial of service application where we captured provenance in real time. The experiment runs an Orbit Traffic Generator (OTG2) on seven WiMAX nodes, and one Orbit Traffic Receiver (OTR2) on one WiMAX node. WiMAX is a new local-area network technology that provides wireless network access within a relatively short range to a WiMAX base station. This is illustrated in Figure 8.

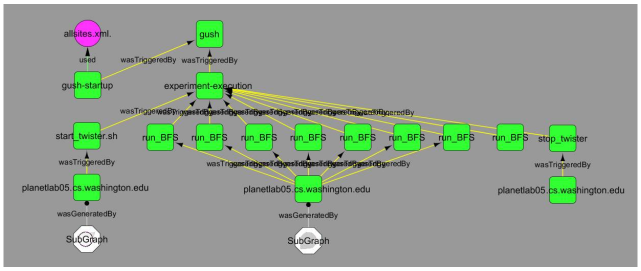


Fig. 7. Abstract view of graph coloring application. Subgraph nodes (shown as two octagons at bottom) result from clustering neighbor nodes; the user can navigate into subgraphs by clicking on them.

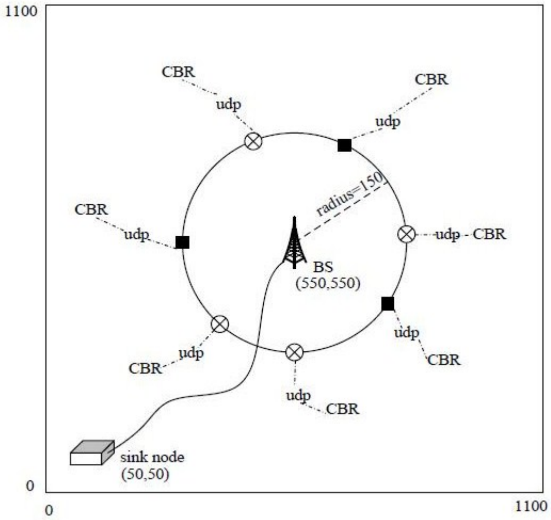


Fig. 8. Network topology in NS2 WiMAX DDoS experiment [5]

The exploratory visualization of this experiment enables interactive analysis of the impact of different parameter configurations. Researchers can see the configuration of WiMAX nodes and the base station, and packets sent and received during the experiment. Packets generated are color coded for each of the 7 sender nodes, and so are the packets received at the base station.The initial view (Figure 9) shows that the WiMAX nodes generated significantly different traffic levels as depicted by the size of their node clusters. By inspecting the packets received at the receiver node (Figure 10), we can discover that only two senders can successfully send packets to the receiver. Finally, by zooming in and comparing the metadata harvested about each sender (Figure 11), we can see that the difference in configuration of ORBIT Traffic Generator (OTG2) results in visibly different throughput results.

VI. Explanatory Data Visualization

By contrast, explanatory data visualization focus on filtering out irrelevant and distracting information to better expose or emphasize the information we want to show. We will show how we use our visualization tool to illustrate the message passing in distributed network experiment, and to provide visual support to conclusions made by other researchers.

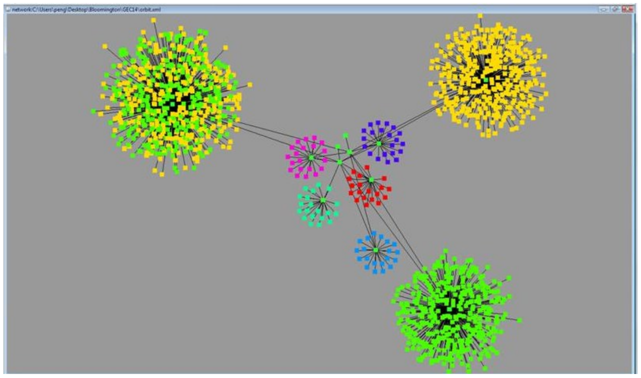


Fig. 9. Initial view under forced-directed layout

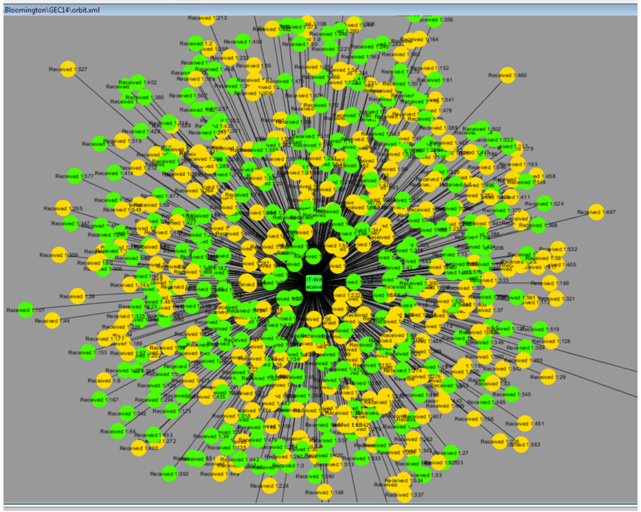
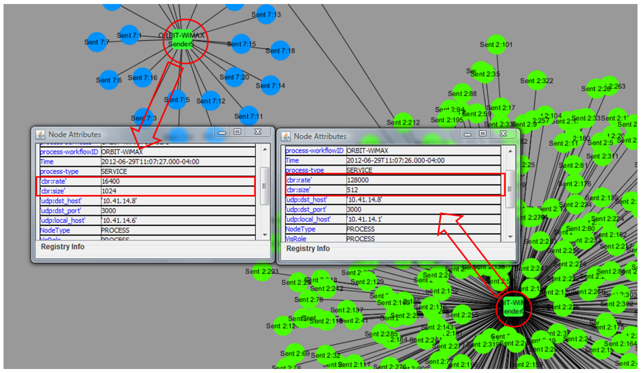


Fig. 10. Packet traffic at the receiver node. Packets are colored according to source IP address. Shows that traffic from only two of the nodes (yellow and green) was successfully received



process-workflowID	ORBIT-WiMAX
Time	2012-06-29T11:07:27.000-04:00
process-type	SERVICE
'cbr:rate'	16400
'cbr:size'	1024
'udp:dst_host'	'10.41.14.8'
'udp:dst_port'	3000
'udp:local_host'	'10.41.14.6'
NodeType	PROCESS

process-workflowID	ORBIT-WiMAX
Time	2012-06-29T11:07:26.000-04:00
process-type	SERVICE
'cbr:rate'	128000
'cbr:size'	512
'udp:dst_host'	'10.41.14.8'
'udp:dst_port'	3000
'udp:local_host'	'10.41.14.1'
NodeType	PROCESS
VisRole	PROCESS

Fig. 11. Comparing attributes of two senders reveals the only difference in parameter related to packet ratio and size

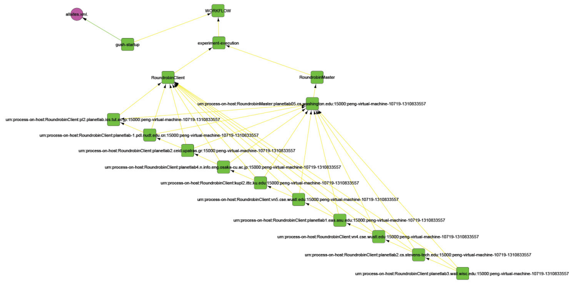

allsites.xml.
WORKFLOW
gush-startup
experiment-execution
RoundrobinClient
RoundrobinMaster
urn:process-on-host:RoundrobinMaster:planetlab05.cs.washington.edu:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:pl2.planetlab.ics.tut.ac.jp:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:planetlab-1.pdl.nudt.edu.cn:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:planetlab2.ceid.upatras.gr:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:planetlab4.n.info.eng.osaka-cu.ac.jp:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:kupl2.ittc.ku.edu:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:vn5.cse.wustl.edu:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:planetlab1.eas.asu.edu:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:vn4.cse.wustl.edu:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:planetlab2.cs.stevens-tech.edu:15000:peng-virtual-machine-10719-1310833557
urn:process-on-host:RoundrobinClient:planetlab3.wail.wisc.edu:15000:peng-virtual-machine-10719-1310833557

Fig. 12. Provenance visualization of Round Robin experiment. Slave node processes are connected by message passing edge to form the slanting line, and the master node is at the up-right area to the line.

A. Round Robin Message Passing

The Round Robin experiment is used to measure the constants for latency and bandwidth between GENI PlanetLab nodes. It has two phases: in phase one, ten slave nodes registered themselves to one master node; in phase two, master nodes sends out a message that is passed to slave node in round robin fashion. We can calculate the message travel time and use it as an upper bound on the messaging network. The provenance graph of this experiment, as shown in Figure 12, renders each node as a host process and each message passing as an edge between host processes. Users can click on the host node to see its configuration information, and its edge to see the information of each message passing.

B. WiMAX Denial of Service

In this experiment we capture and visualize provenance from a denial of service experiment running in the WiMAX network. The experiment is run on the NS2 simulator [11]. The experiment, called DoS Attacks Exploiting WiMAX System Parameters [5] uses 100 subscriber stations with varied configurations of 6 parameters running on NS2 (Figure 8). We apply our visualization techniques to represent critical provenance regarding packets that were dropped, and by doing so are able to convey information about DDoS attacks through visualization. Previously the researchers were not able to glean the kinds of behavior except by post-mortem statistical analysis on measurement data. In our visualizations, the Base Station (BS) is displayed at the center, and all Subscriber Stations (SS) are displayed in the large circle, surrounded by a small circle of their dropped packets (red) and/or received packets (blue). Normal user SS is connected with yellow edge, while attacker SS is linked via red edge. Figure 13 shows the zoomed in visualization of SS, where you can also see the overview picture at the left bottom corner.

The visualization techniques have special layout algorithms and visual style that enable side-by-side performance comparison of different experiment configurations. The 3x3 visualizations in Figure 14 show packets both dropped and received. The visualization is adjusted automatically for the provenance data volume based on total number of packets sent, under the layout based on network topology. Each row of graphs correspond to one row in the table above the visualization: the left graph shows both the dropped and received packets, middle graph shows the dropped packets only and the right graph shows received packets only.

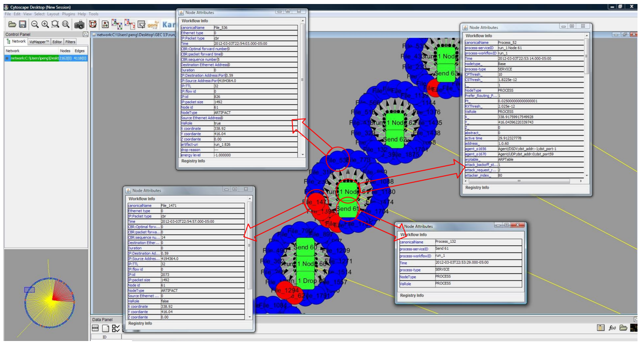


Fig. 13. Zoomed in visualization shows detailed attributes of each dropped packet, delivered packet, receiver node, and process.

Run id	Frame duration	number of attackers	attack backoff start	attack request retry	bw backoff start	bw request retry
1	0.004	20/80	1	2	1	2
244	0.01	20/80	1	2	1	2
487	0.02	20/80	1	2	1	2

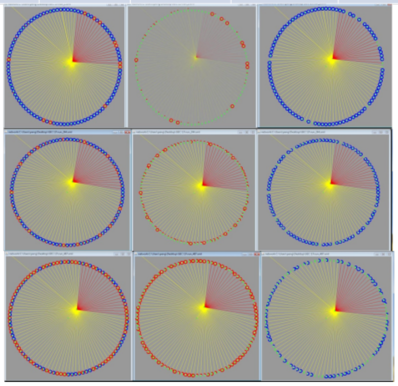


Fig. 14. Packets dropped increases as frame_duration increases from 0.004s to 0.02s.

VII. Conclusion and Future Work

We developed visualization techniques for exploratory and explanatory uses of provenance. The techniques are demonstrated through two case studies, one a provenance capture of a satellite imagery ingest processing pipeline, and the other a provenance capture of the network layers of large-scale distributed network applications. The case studies demonstrate the power of the visualization tool. The clover leaf history chain visualization illuminates dependencies between daily workflows in a way that was not available before. The packet drop/packet sent visualizations illuminated to the Clemson University researchers aspects of their denial of service experiments in a way they had not been able to see before.

The AMSR-E pipeline project illuminated the need for better process-level transparency. Occasionally the science algorithms in the AMSR-E pipeline would be updated, but there was no way for the algorithms to be queried to produce information about themselves. That is, once embedded in the processing pipeline, they became part of the processing engine. Researchers are not interested in the repeatable details about which housekeeping algorithm was run, they were interested in which version of a science algorithm was used, and this latter information had been buried away with the code.

We continue to improve the efficiency of provenance representation and are examining graph databases for faster query time on provenance graphs. We are working on visualizations of other abstract representations, such as a temporal representation that uses Lamport clocks. We are also improving the efficiency of the graph matching algorithm for larger cases than just comparison of two graphs.

Acknowledgment

The authors thank Richard R Brooks, Clemson University and Helen Conover, University of Alabama Huntsville for their help in the use cases.

References

[1] R. Bose and J. Frew, "Composing lineage metadata with xml for custom satellite-derived data products," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE, 2004, pp. 275–284.

[2] K. Cheung and J. Hunter, "Provenance explorer–customized provenance views using semantic inferencing," *The Semantic Web-ISWC 2006*, pp. 215–227, 2006.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[4] N. Del Rio and P. Da Silva, "Probe-it!: visualization support for provenance," in *Proceedings of the 3rd international conference on Advances in visual computing-Volume Part II*. Springer-Verlag, 2007, pp. 732–741.

[5] J. Deng, R. Brooks, and J. Martin, "Assessing the effect of wimax system parameter settings on mac-level local dos vulnerability," *International Journal of Performability Engineering*, vol. 8, no. 2, p. 183, 2012.

[6] F. DePiero, M. Trivedi, and S. Serbin, "Graph matching using a direct classification of node attendance," *Pattern Recognition*, vol. 29, no. 6, pp. 1031–1048, 1996.

[7] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, "Twister: a runtime for iterative mapreduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 810–818.

[8] C. Elliott, "Geni–global environment for network innovations," in *33rd IEEE Conference on Local Computer Networks*, 2008, p. 8.

[9] J. Freire, C. Silva, S. Callahan, E. Santos, C. Scheidegger, and H. Vo, "Managing rapidly-evolving scientific workflows," *Provenance and Annotation of Data*, pp. 10–18, 2006.

[10] N. Iliinsky and J. Steele, *Designing Data Visualizations, Intentional Communication from Data to Display*. O'Reilly Media, 2011, ch. Chapter 1: Classifications of Visualizations.

[11] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*. Springer Verlag, 2011.

[12] M. Kunde, H. Bergmeyer, and A. Schreiber, "Requirements for a provenance visualization component," *Provenance and Annotation of Data and Processes*, pp. 241–252, 2008.

[13] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers *et al.*, "The open provenance model core specification (v1. 1)," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743–756, 2011.

[14] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, B. Didier, N. Ashish, and C. Goble, "Multi-scale science, supporting emerging practice with semantically derived provenance," in *ISWC workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*. Florida, 2003.

[15] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 2004.

[16] P. Shannon, A. Markiel, O. Ozier, N. Baliga, J. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[17] Y. Simmhan, B. Plale, and D. Gannon, “A survey of data provenance in e-science,” *ACM Sigmod Record*, vol. 34, no. 3, pp. 31–36, 2005.

[18] ——, “Towards a quality model for effective data selection in collaboratories,” in *Workshop on Workflow and Data Flow for Scientific Applications (SciFlow06), held in conjunction with ICDE*. IEEE, 2006, pp. 72–72.

[19] ——, “Karma2: Provenance management for data-driven workflows,” *International Journal of Web Services Research (IJWSR)*, vol. 5, no. 2, pp. 1–22, 2008.

[20] J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, and M. Greenwood, “Using semantic web technologies for representing e-science provenance,” *The Semantic Web–ISWC 2004*, pp. 92–106, 2004.