Tapestry Architecture

Sandhya Narayan
Infoblox Inc.
[work in progress]

Introduction

Tapestry is a system which extracts the “network complexity” (NCI) of an enterprise by examining the DNS requests made on that network. NCI can be viewed either hourly, weekly, monthly, annual and any other period based on selection.

Figure 1 shows the Tapestry UI with a graph of the “Network Complexity Indicator” over time, and a dashboard showing some performance numbers.

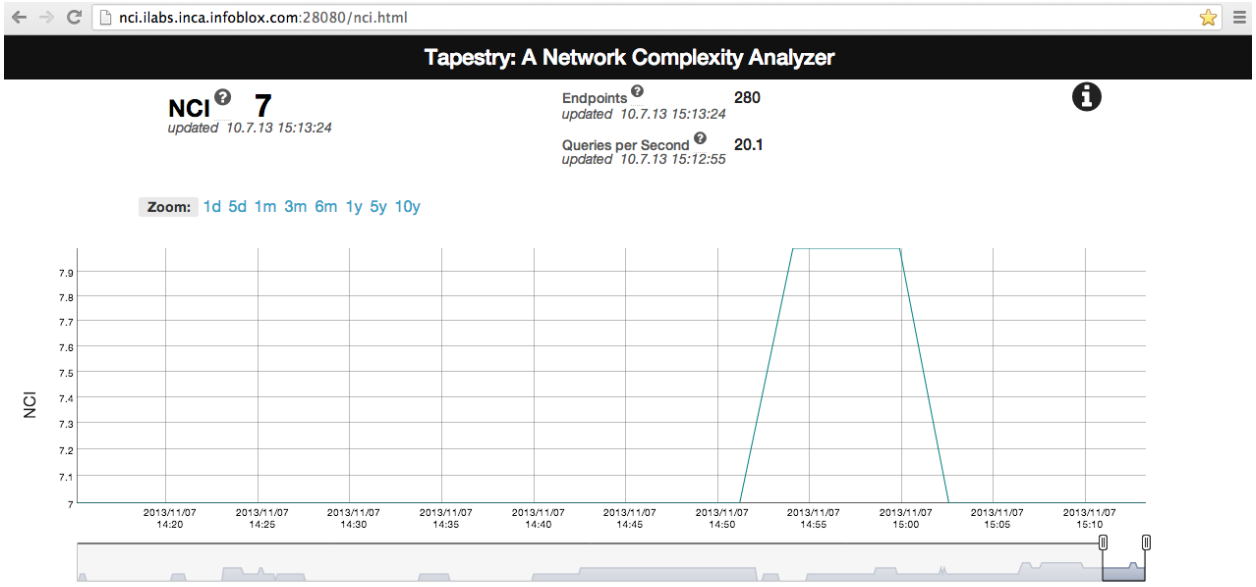


Figure 1

Tapestry SDN application

Tapestry data is collected by a SDN application which utilizes an OpenFlow switch as a DNS data collector. In our example architecture, we use a Network White Box (NWB) that runs LINC OpenFlow switch software as the OpenFlow switch. These NWBs (also called Tapestry Collectors) are placed in-line to one or more DNS servers and tap DNS responses being sent to DNS clients. The Tapestry Collectors send the collected data to the Tapestry system (Figure 2) for analysis, which then provides instantaneous NCI values and a set of other indicators computed over selectable duration.

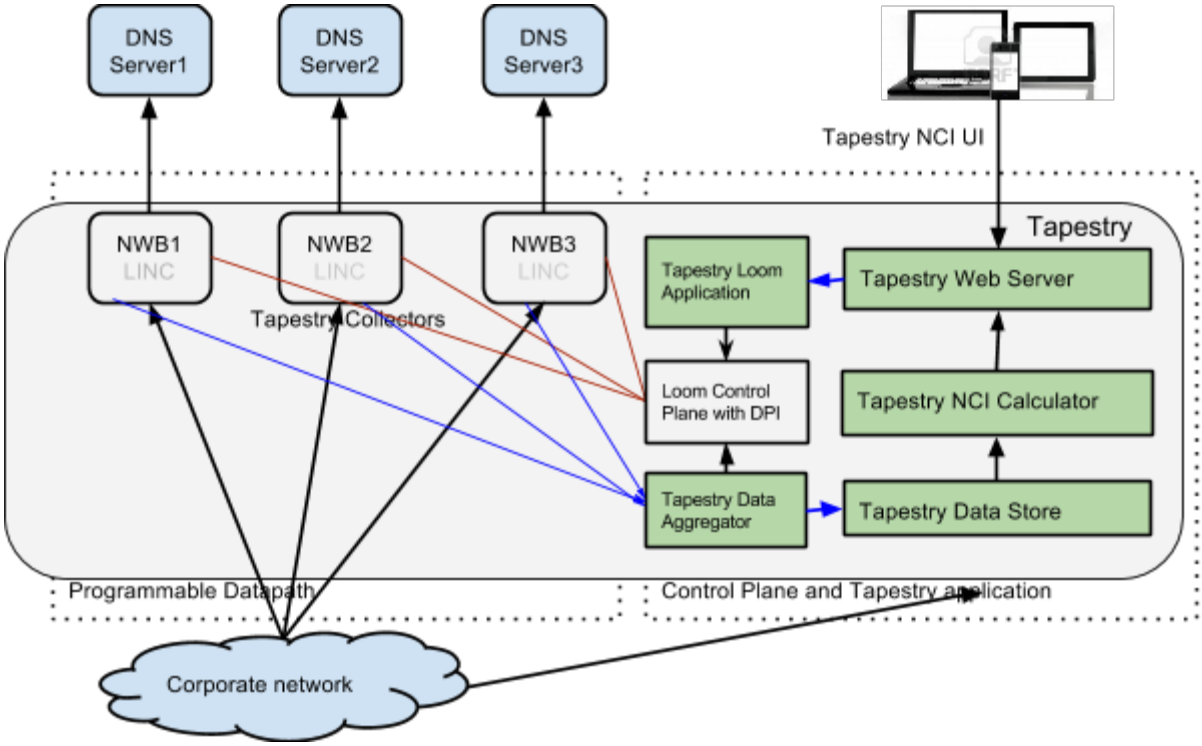


Figure 2

Figure 2 shows the components of the Tapestry system listed below and their interconnections. The next section provides more details about these components.

- Tapestry NCI Calculator
- Tapestry Web Server
- Tapestry Network Complexity UI
- Tapestry Collectors
- Tapestry Loom Application
- Tapestry Application
- Tapestry Data Store
- Tapestry Data Aggregator
- Loom Control Plane

Tapestry using DNS logs

Instead of using DNS data being collected by OpenFlow switches, Tapestry can make use of DNS log data made available by InfoBlox DDI appliances to an FTP server. In this case, Infoblox DDI appliances send DNS log data to the Tapestry FTP server, which sends it directly to the Tapestry Data Aggregator where it is filtered and parsed, making it suitable to be used by

Tapestry for NCI analysis. Figure 3 shows the components of Tapestry for this type of deployment.

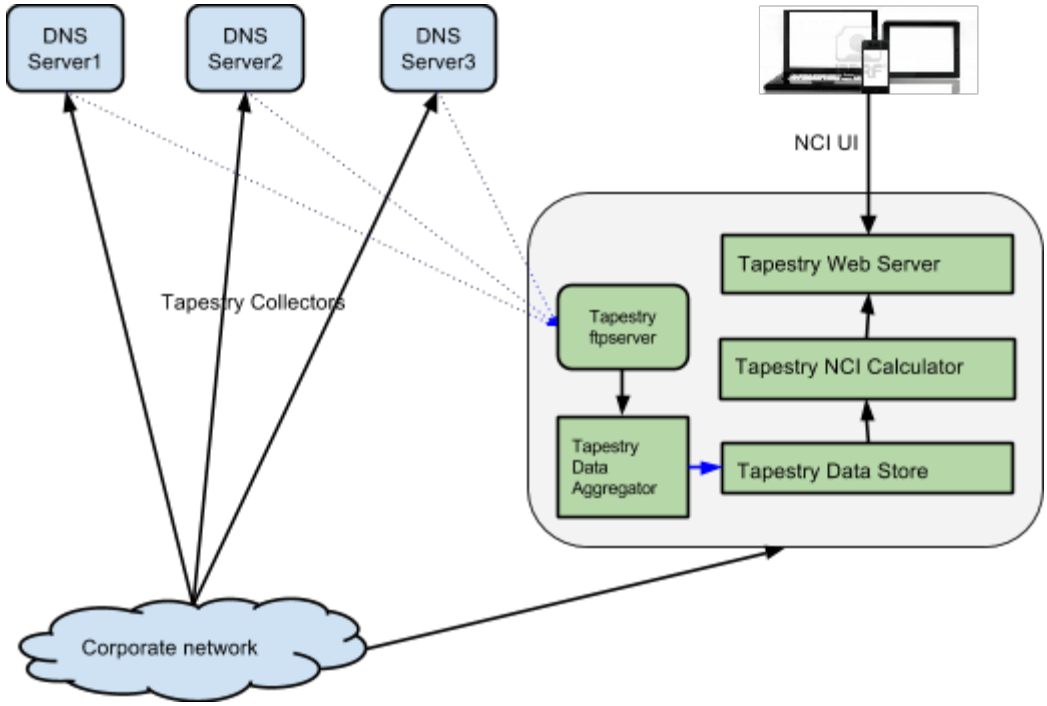


Figure 3

Note: The rest of this document will focus on the Tapestry SDN application.

Tapestry Components

Figure 2 shows how Tapestry data collection is done using a distributed SDN application that utilizes OpenFlow switches and control plane. Other Tapestry components aggregate the data, store it, compute NCI and display it on different types of clients such as web clients, smartphone clients and iPad clients. This section describes these components.

Tapestry NCI Calculator

The details of the NCI algorithm are discussed in http://www.flowforwarding.org/nci-article. In summary, a network's complexity can be judged from the interconnections in the network between different end-points, their interactions and information exchanged. Positive DNS responses serve as a simple but good indicator for interactions between end-points. The NCI is a single number computed from pairs consisting of {DNS requesting client's IP address, IP address resolved by DNS}.

Tapestry Web Server

Tapestry Web server facilitates the users to view Network Complexity Index in real time or for any chosen period. It is currently implemented using Yaws, a HTTP high perfomance 1.1

webserver written in Erlang, particularly well suited for dynamic-content web applications (http://hyber.org/). YAWS supports the WebSocket Protocol, which enables two-way communication between clients and web servers. Tapestry Web Server and client use WebSockets for interaction with each other.

Tapestry NCI UI

Tapestry NCI UI allows the user to select the data for analysis by specifying the following: fields of DNS data, range and type of DNS entries. The Tapestry NCI UI is a web client that uses html, css and javascript in a single page. It uses WebSockets to communicate with the server and asks the Tapestry server for the following: a) start pushing fresh updates and b) select time/duration of interest. The server sends NCI, end-points and query rate information for display. The Tapestry NCI UI displays these values putting NCI data on a graph. Figure 1 shows an example screenshot of the Tapestry web client.

Tapestry Collectors

OpenFlow switches placed in-line on the path to DNS servers and serve as Tapestry Collectors. They tap DNS responses and send them to Tapestry Data Aggregator. Let us consider an example NWB with four network interfaces and running LINC OpenFlow switch software. Port 1 is connected to the DNS server and Port 2 is connected to the corporate network.

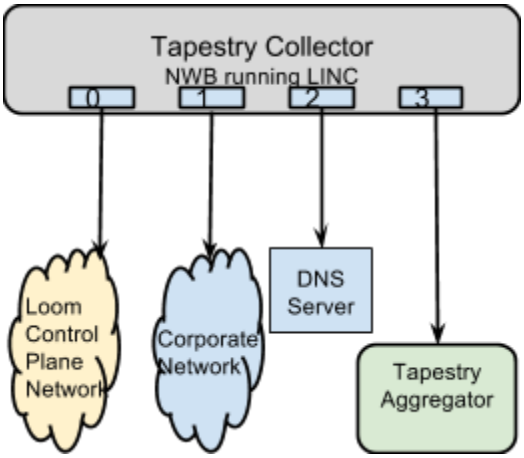
There are several ways to collect the DNS traffic:

1. Direct connection: In the simplest case, copy DNS response packets to Port 3 (not used otherwise) and directly connect a cable from Port 3 to the Tapestry Data Aggregator.
2. Use of Packet-In: Send DNS response packets as Packet-in messages to the OpenFlow controller, which sends it to the Tapestry Data Aggregator.
3. Use of Packet-in with in-band control: Same as (2) but make use of "In-band" controller.

These options are discussed below:

Direct Connection

In the figure below, Port 3 of the switch is the "tap" port and is connected directly to the Tapestry Data Aggregator. This configuration is the simplest to implement, but requires new wiring to connect the "tap" port to the Aggregator.



Flow entries to be set up in the OpenFlow switch:

- Higher priority: Forward UDP traffic from port 2 to port 1 and port 3 (tap DNS responses).
- Lower priority: Forward traffic from port 2 to port 1 (forward the rest of the traffic)
- Lower priority: Forward traffic from port 1 to port 2 (forward all the traffic)

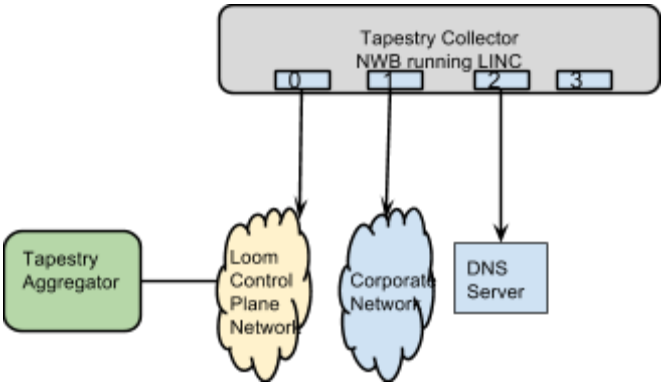
Use of Packet-in

This method uses “packet-in on action” feature of OpenFlow. A switch sends a packet-in message to the controller when there is a match with a flow entry for an incoming packet and the action-output port for the matched flow-entry is set as “controller”. The switch encapsulates the incoming packet in the packet-in message. The controller decapsulates the packet-in message and sends the data (the incoming UDP packet) to the Tapestry Data Aggregator.

Flow entries for Packet-In:

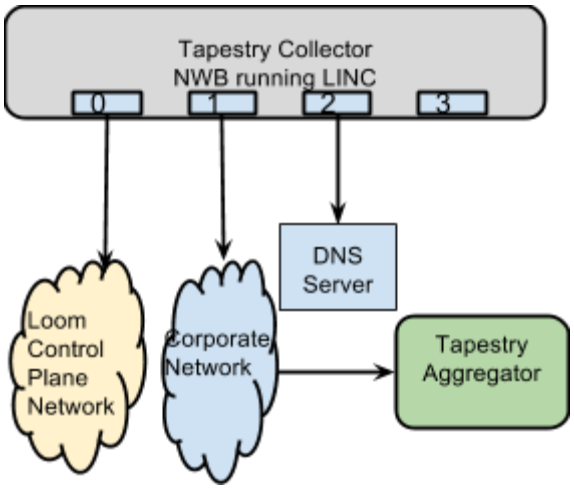
- Higher priority: Forward UDP traffic from port 2 to port 1 and controller (tap DNS responses).
- Lower priority: Forward traffic from port 2 to port 1 (forward the rest of the traffic)
- Lower priority: Forward traffic from port 1 to port 2 (forward all the traffic)

The advantage of this method is that a packet-in message encapsulates the “matched” packet and so the entire packet can be captured at the Aggregator. It also has the obvious advantage of not needing any extra cabling to connect the tap port to the Aggregator. It also has the advantage of using TCP to send the “tapped” packet, because of the use of OpenFlow to collect “tapped” packets.



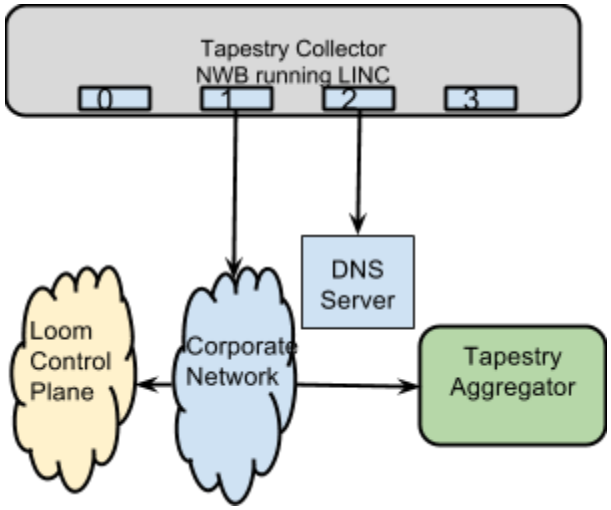
Change Destination IP of the “tapped” packet

This method sends the “tapped” packet to the Tapestry Data Aggregator via the Corporate network by rewriting the destination IP address in the header and outputting the packet to the port connected to the Legacy Network. It has the obvious advantage of not needing any extra cabling to connect the tap port to the Aggregator as in the direct connection method. But it loses the actual destination IP address and port in the header. It it to be noted that the “tapped” packet is a udp packet because original packet is a udp packet.



Packet-In message with In-Band Controller

In this method the control channel is in-band and not connected to a different port on the switch. Otherwise the method is the same as the use of packet-in.



Tapestry Loom Application

Tapestry Loom application prepares hand-coded flow entries for all the LINC OpenFlow switches and uses the Loom Control Plane to send them to the switches.

Tapestry Application

This is the main application which starts the other components of the Tapestry system. The current design assumes the use of packet-in messages to collect DNS responses. To capture the DNS responses this application starts a secondary OpenFlow controller whose only purpose is to capture packet-in messages, decapsulate them and perform DPI to identify the pairs of the form: {DNS requesting client's IP address, IP address resolved by DNS}.

Tapestry Data Aggregator

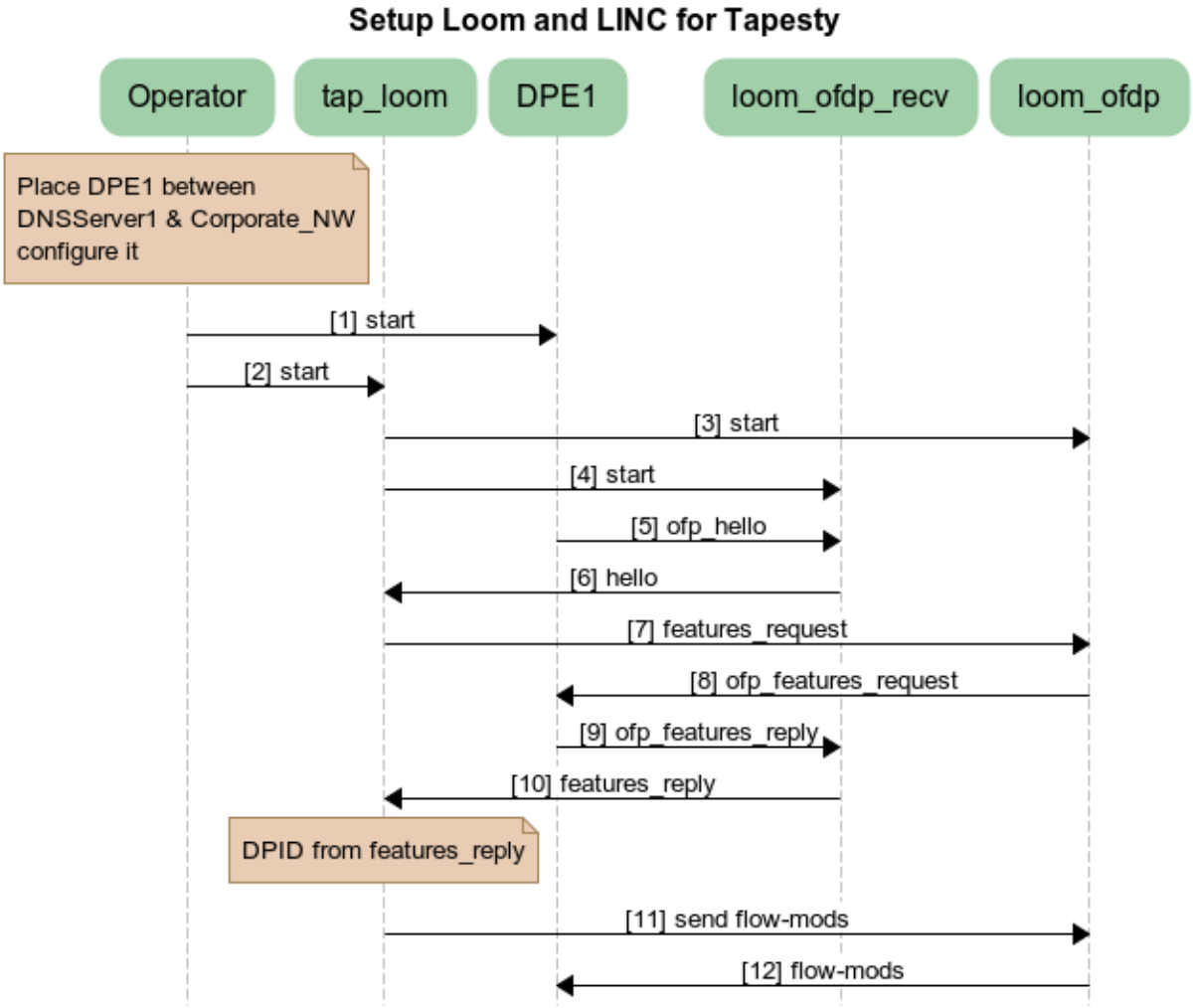
Tapestry Data Aggregator subscribes to the Secondary OpenFlow controller started by the Tapestry application and gathers the pre-processed DNS response pairs and stores it in the Tapestry Data Store.

Tapestry Data Store

The Tapestry Data Store stores the data collected and aggregated by the Tapestry Data Aggregator. The DNS response pairs {V1, V2} are stored as directed edges in a graph database where V1 and V2 are vertices. Over time as new edges are formed, the graph database captures a view of all the dynamic interconnections between systems. This serves as an input to the NCI Calculator.

Tapestry Interactions

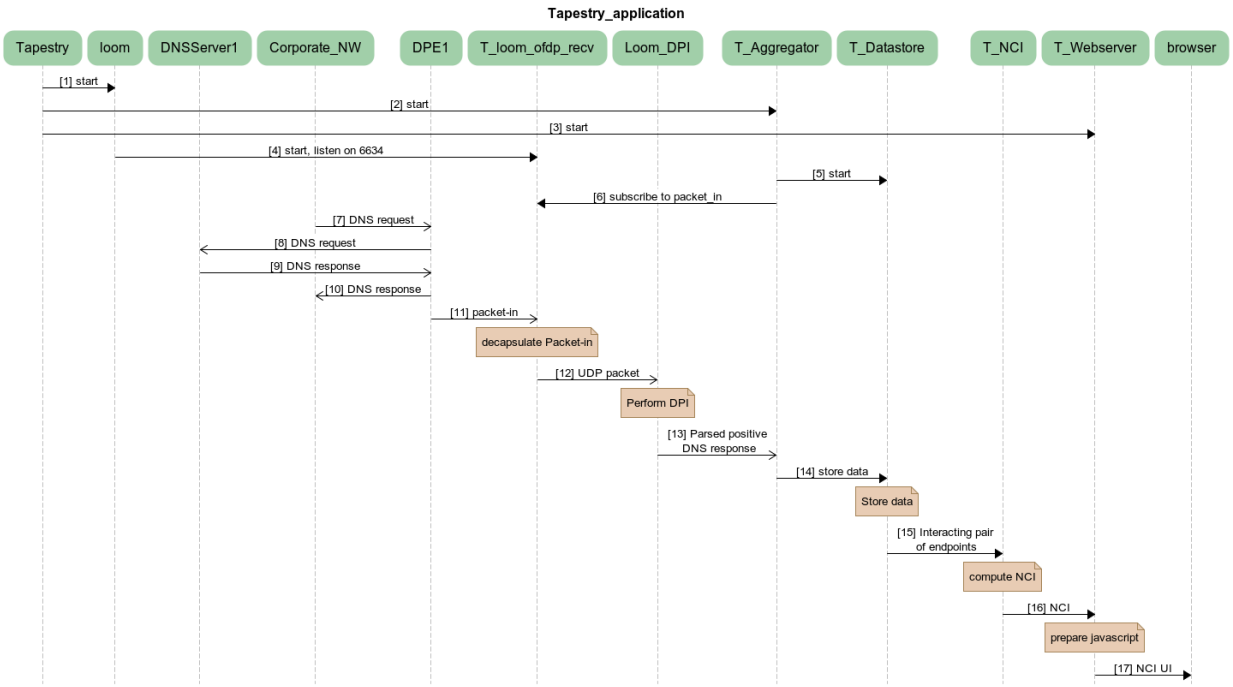
The sequence diagrams below shows the interactions between various components in the Tapestry system. The setup sequence shows interactions between the SDN components of Tapestry as the components are set up for running Tapestry. The second diagram shows the interactions that occur as DNS requests and responses are forwarded by the DPE.



1. Place switch (DPE1) between DNSServer1 and the corporate network. Configure swtich to connect to a master controller at {MasterIP, MasterPort}, and a slave controller ar {SlaveIP, SlavePort}. Start the switch.
2. Start tap_loom application, which starts the master controller.
3. tap_loom starts the sender (loom_ofdp)
4. tap_loom starts the receiver (loom_ofdp_recv)
5. Switch connects to the controller, and sends hello message.
6. Controller (tap_loom) receives hello message, and protocol version of the switch.
7. Controller (tap_loom) sends features_request to loom_ofdp to get switch features.
8. loom_ofdp sends the actual OpenFlow message to the switch.
9. Switch sends features_reply.
10. loom_ofdp_recv sends features_reply to tap_loom. tap_loom finds out the DPID from features_reply and uses it to identify all interactions and state related to the that switch.

11. tap_loom hand-codes the flow-mod messages as described in the Tapestry Collector section and sends them to the sender loom_ofdp.
12. loom_ofdp sends the OpenFlow messages to the switch.

Note: The sequence described above is close but not exactly the same as the actual implementation.



1. Start Tapestry application.
2. Tapestry starts loom as a slave controller
3. Tapestry starts Tapestry Aggregator
4. Tapestry starts Tapestry Webserver
5. loom starts receiver T_loom_ofdp_recv to listen on port SlavePort
6. Tapestry Aggregator starts Tapestry Datastore
7. Tapestry aggregator subscribes to
8. DNS request made by some client arrives on Port2 of DPE1
9. DNS request is forwarded to Port1 of DPE1
10. DNS response arrives at Port2 of DPE1
11. DNS response is forwarded to Port1
12. Packet-in Openflow message is sent to the controller
13. Packet-in is decapsulated to get UDP packet (DNS response)
14. DNS response is parsed to get endpoints of positive responses
15. {DNS requestor IP, resolved IP} are stored in Datastore
16. Set of endpoint pairs are selected and NCI is calculated
17. NCI is sent to Tapestry Webserver

18. Browser client is updated with changing NCI value

Tapestry Requirements

Datapath

- Programmable Ethernet Datapath supporting OpenFlow 1.3
- Depends on the environment of deployment.

Some example hardware requirements:

- Low cost low power NWB
    - Intel Atom N450 1.66GHz based desktop platform with 4GbE: http://www.portwell.com/products/detail.asp?CUSTCHAR1=CAD-0205-06-08
- Medium cost medium power NWB
- Soft switches on any x86 server

Loom Control Plane

- Depends on the environment of deployment.
- x86 running Linux or Windows

Tapestry Application

- Depends on the environment of deployment.
- x86 running Linux or Windows

Glossary

Network White Box (NWB): An X86 based system with at least 3 network interfaces, preferably with more cores than interfaces.

LINC: A software OpenFlow switch supporting OpenFlow 1.2 and 1.3.1 specifications written in Erlang.

Loom: A distributed controller platform for SDN supporting OpenFlow 1.0, 1.2 and 1.3 specifications.