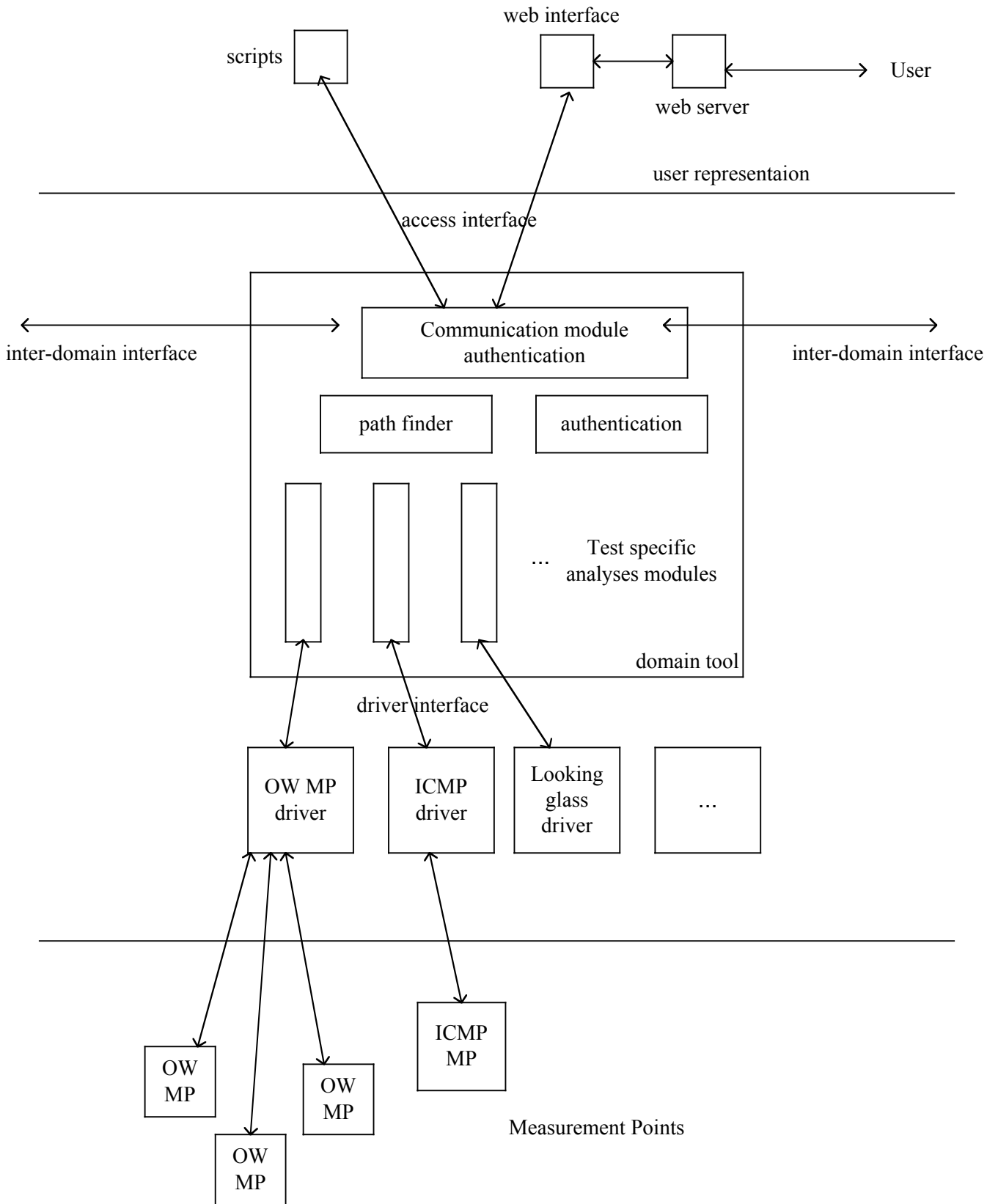


Performance Monitoring – Domain Tool – Architecture Requirement 0.1

The general goal of the “*domain tool*” is to provide a view of network performances across different domains by performing actions across these domains. The actions can be: perform a test between two different domains (without an remote human intervention in another domain), retrieve monitored data from a test, perform several looking glass like actions in different domains.

The inter-domain performance tool consists of several parts: the *Domain tool*, the *web interface* and the *drivers*.

The core of the tool, namely the *domain tool*, is mainly in charge of treating the data retrieved, by the *drivers*, from the Measurement Point (MP) according to the user request. It is also in charge of the communication with other *Domain tools* (via the *inter-domain interface*) and the *web interface* or *scripts* (via the *access interface*). The drivers are in charge of the communication between the domain tool and the Measurement Points.



The web interface

The *web interface* acts as an interface between the user and the domain tool. It is behind a web server allowing the user to request a test, see the result of a test, etc. The *web interface* can be php, servlet, cgi, asp, etc) Some additional analyses can be done via some *scripts* according to the needs of the domain (cgi, shell, etc).

The *scripts* and *web interface* representation tool is developed by the domain according to its own needs. E.g combination of several plots or several metrics on a same page or the same information displayed with different aggregation functions (average and percentiles). Or to perform operations on the data that are not provided by the *domain tool*.

Note: On the long term, the functionality developed can be moved within the *domain tool*.

The only requirement on the representation tool is to inter-act with the *domain tool* via the defined *access interface*.

It should be able to find a local or closest *domain tool*.

Domain tool

The *domain tool* is at the hart of the system. That's where the requests coming from another domain/users are treated accordingly to the user requirements. It knows which test can be run on the domain, what aggregation function on the test results it is capable of doing, it can find useful information about the other domain tools, etc

It implements the following functionalities:

- Authorisation
Based on the information provided by the user and on agreement between domains, it allows the user to have access to a defined set of functionality's (action, metric, aggregation function).
- Authentication
Based on information provided by the user and on the agreement between domains, it allows a domain tool to identify the other domain tool and user with which it communicates.
- Path Finder
The Path finder module has several functions, all related to finding either a measurement point inside the domain or the next *domain tool* to be contacted or the *domain tool* the closest to an IP address.
The path finder module is in charge of finding the Measurement Point inside its own domain the closest to a given IP address, the Measurement Point the closest to an domain egress point and the measurement points along a given path. The Path Finder also is in charge of finding the next *domain tool* towards an IP address or the *domain tool* the closest to IP address.

- Library of aggregation functions
The *domain tool* should also contain a library of aggregation function (an aggregation function being: mean, percentiles, standard variation, etc). An aggregation function can be applied on different metric. The goal is to re-use as much as possible existing function. New aggregation function easily be added should.
- Data analyses
When the domain tool receive the sets of data (either data from the domain via the drivers or from another domain via the *domain tool to domain tool interface*), it has to apply a proper aggregation function to this set of data according to the type of aggregation the user has requested. It may also have to perform some specific actions on the data received from the other domains involved in the test (eg add the averages, to provide a single value to the user, etc).
- Documentation
Information should be provided to developers on how to integrate something new, how to write a driver, etc
- Tool administrator
How can he access the tool, what can he changes (list of MPs, their type, authorised users and what they can access, the other domain tools -?-) etc
- Treatment of exceptions
When a request is done by a user, feedback should be sent if an action cannot be performed.
- OS independent
- Find other domain tools
The tool should automatically find the other domains tools.
- The tool should also provide the user or another domain tool with the list of test, which can be performed on the domain and with which parameters.
- Easy to expand for some new test
The domain tool should be build to easily cope with extension to new type of tests, type of aggregation, new type of functionality's for the path finder, new drivers, new type of Measurement Points. A new type of test should be easily added. (Think about it a bit as object oriented programming)
- the tool should be aware of the test, the metric, the metric parameters and the aggregation function which can be performed in the domain.
- access interface
The interface between the domain tool and the representation tool specify the actions which can be requested to the domain
 - request to start a test
 - to get data
 - to provide the capability of the domain and/or a given type of test
 - exceptions signalling

The interface should easily be expandable to new requests (add new metric, new test, new aggregation request)

- Domain tool – domain tool interface
The interface between the domain tool and another domain tool specify the actions which can be requested between domains
 - request to start a test
 - to get data
 - to provide the capability of the domain and/or a given type of test
 - exceptions signalling

It should easily be expandable to new requests (add new metric, new test, new aggregation request).

Preferably, we need to work with a group as GGF forum to have an interface general enough and usable by a lot of people.

Drivers

A *driver* is specifically build for a Measurement Point type (eg a one-way MP or a router). It is in charge of the communication between the domain tool and the measurement points it is in charge of. It configures the MP to start and to finish a test. The *driver* is also in charge of retrieving the data from the MP or from the database associated a particular tool (the data can be store in a database elsewhere than on the MP).

The *driver* has also to do some resource management for the MPs in order to avoid a too heavy utilisation of the MP, which could end up in disturbing the measurements. Some utilisation guidelines have to be provided for each type of MPs (they should be provided to the *driver* as configurable parameters, eg amount of packet per second which can be sent from/to a given MP).

One have to keep in mind that a MP could handle different type of tests. A measurement point can be a one-way measurement box or a router or a workstation. It is currently left open if one driver is in charge of a MP type (and so of all the test which can be run on this MP) or if one driver is in charge of one type of test on a specific type of MP. Both solution have some advantages (a driver per MP type is advantageous for resource management, whilst a driver per test on a MP type is advantageous for re-usability between domains).

A test can produces different metrics (eg a OW test can provide some information about the delay, jitter, packet loss).

The *driver* should also announce to the *domain tool* the capabilities of the domain for the test it is in charge of (parameters configurable, MP location, resources management capabilities, etc)

The *driver* has also to cope with the treatment of exceptions and pass them to the *domain tool*.

General remark

- we have to define how to cope with un-instrumented domains. An un-instrumented domain is a domain without *domain tool*.
- each domain can deploy its own monitoring infrastructure. There is a strong need to have the equipment inter-operable if some tests are performed between domains having deployed different type of equipment.
- The “end-to-end” statistics available depends on the type of test each domain is capable of performing, e.g. when a domain is not able of performing one of the test.
- Development of an interface between the *domain tool* and the “*user-representation*” *tool*. This interface will be based on XML technologies and reflects the services the users can get from the domain.
- It is not yet clear if the web-interface will only connect with its local *domain tool* or if it can connect with other domain tool (and be oriented by the local *domain tool*)
- Chaining vs reference.
-

We are starting the tool with a very restricted set of test (as proof of concept phase). The first set of actions will be based on one-way measurement tools (OWD, IPDV, OWPL, re-ordering, traceroute).

This came out of the discussion of the performance monitoring held in Dublin the 31/03 and 1//4/03.

Attendance:

Simon Leinen – SWITCH
 Ioannis Kappas – DANTE
 Victor Reijs – HEANET
 Nicolas Simar – DANTE
 Dave Wilson – HEANET
 Chris Welti – SWITCH

ACTIONS:

- SL and VR to come with the specification of the *Domain tool – representation tool interface* and the *Domain tool – domain tool interface*. The specification has to be done by Mid June to allow its implementation by September. Contact GGF forum.
- Loukkik to start implementing the tool starting from the device for RIPE box then the other main functionality for the tool (can use the SWITCH and HEANET ones).
- NS to contact Henk to ask how can we access and retrieve the data from a RIPE box.
- CW and SL to provide guidance to the group about the *Path Finder* module, then to implement it. Milestone: implementation should be complete by September for the trial. Group to answer to these questions.
- Dante has to glue together the pieces developed.
- Dante can provide some effort on implementing the communication module.