Bringing federation one step beyond through MySlice

Abstract. In this article, we introduce MySlice, which started as both a web-based GUI and an API to support users in selecting resources polled from the SFA federation through different Aggregate Manager APIs. It also connects to TopHat API for measurements, and to the –MyPLC or– SFA registry API for authentication and user management purposes. MySlice abstracts the results of all these API calls, and merge them into a single format that can be queried by the user transparently, –?–with caring about the source. We designed MySlice around a fully modular design, which allows developers to build plugins for the display of informations, and customize user interactions with data; a set of wrappers allow to communicate with different APIs. We believe that MySlice offers a solid and convenient framework that handles complex details for developers willing to extend it. It thus allows to build a portal integrating various independent services transparenly, which might be useful to offer users a single entry point to the federation, and bring them to different testbeds and tools required for their experiments.

1 How far is a testbed federation from a global experimental facility ?

Testbed federation has proven an efficient and cost-effective way to experiment with what is envisaged to be the Future Internet. It is indeed conceivable today to imagine an ecosystem involving multiple actors, each providing a set of resources eventually virtualized, allowing the coexistance of overlays realizing multiple network paradigms.

There seems to be an emerging consensus, especially within the FIRE and GENI communities, towards the adoption of the Slice-based Facility Architecture (SFA). It has the noticeable proven record of running a global federation already involving a dozen of testbeds worldwide, such as the different PlanetLab facilities. This has led to the development of a large amount of tools and services whose objective is to support users in various tasks related to experimenting on the platforms, and that are complementary to SFA. We could roughly categorize them as: 1) user and authentication management [1]; 2) testbed resources browsing and booking [2, 3]; 3) policies [4]; 4) resource scheduling; 5) measurements [5]; and 6) stitching.

A user willing to run a non-trivial experiment will face the issue to contact all those different sources, with the associated complexity we can imagine, and it will have to run through and make sense of a large amount of data. Even in the favourable context of SFA, where we are converging to a consensus unique API,

we are facing the necessary complexity of a design which allows a secure and distributed thin waist for the federation, and the possible diversity of RSpecs and extensions developed to match the diversity of testbeds.

A user with an experiment in mind does not need to be exposed to the inner details of the federation, and should be guided to different services of interest. –?–This is possible since we are in a different context close to a community of users that will trust the tool, and where we could allow for more centralization.

– current state of the Myslice Platform ? – In this document, we even go a bit further and describe how, with minimal modifications, the current MySlice platform can be extended to form an entry point to the federation for experimenters, present an integrated view of the necessary information, and orchestrate the different interactions to support the experimental lifecycle.

2 Related work

There exist several initiatives proposing integration schemes, but they tend to limits themselves to one specific domain (this is for example the case of semantic frameworks for measurement data). The most elementary user entry points are typically website giving pointers to a set of testbeds and tools (repositories) of interest, with examples such as the OneLab experimental facility website [?], or the related sections in the FIRE and GENI websites [?,?]. [?] proposes a common login feature between the various testbeds and tools it offers. The soon-to-be-released GENI portal goes further by offering an integrated portal for user registration, and a seamless switching with tools of interest (such as Flack) thanks to the use of an SSO solution. We are building on such initiatives, and have generalized the latter approach in MySlice so that it can become an exensible community-base platform on which to build further improvements.

3 Design of the MySlice platform

MySlice architecture is composed of three bricks mapping the essential functionalities and interfaces: (1) a core library (in Python) acts as a broker between the user and the various interconnected systems, and implement more of the functionalities; (2) these are in turn exposed to an XMLRPC API; and (3) a rich and modular web interface features various tool to allow the user to easily navigate and make sense of the available information. Those components are easily extendable through a set of simple API abstracting the developer from the intricate details of the different participating entities.

An interconnection framework This framework is an extension of the solution proposed in [5], which is in use today in TopHat to provide an interconnection of measurement systems. And MySlice builds on it to annotate resources with measurements and other types of data. This framework offers four components essential to an easy and transparent data integration: (1) a simple an efficient query language to request data and execute actions across interconnected

platforms (think as SQL); (2) an engine performing query dispatching and result aggregation; (3) a standard interface for plugging new platform APIs, along with a metadata syntax for describing them; and finally (4) a set of gateways to support platforms not complying to this interface (such as SFA). Those gateways are responsible for handling authorization/authentication with the platforms, and interpreting the transport of data (eg. XMLRPC), their format (eg. XML), and the ontology used for representing information. Note also that both TopHat and MySlice APIs corresponds to this standard interface, and that is closely mimicks the widely use MyPLC API that is the native interface of the PlanetLab testbeds.

A modular interface for browsing and visualization TODO

Plugin-based web user interface Different objects with different properties : eg. resource, type, specific properties we might want to display them in a unique fashion

Different views/presentations, from simple list to some exploiting properties : geographical dimension (lat/log, xyz), temporal, etc. can be very specific to a testbed

or a new functionality like scheduler requiring a user interface, not only to browse data, but also to edit it.

or making sense of the large amount of available information and data in the federation need filtering, grouping, annotating, etc. = expressing queries, in addition to visualization

need to make a choice

help the user formulate a query, cf the mechanism we have just presented: filters, fields, = searching, grouping, sorting, etc. - we know what we want - we guide discovery - we solve from a set of constraints the optimal choice constraints, etc.

answers = simple tables

plugins system - query mechanism make a nice abstraction for all the data - user only need to know the semantic, and they got a nice flat interface to request and obtain data : only thing to know. - only say which information it is interested in, and if available it - no worry about the source - make it easy to add new functionality, and can go well beyond what we have presented : - accounting/tutoring - consistent layout and navigation, modular also - compoenents to build a social layer, etc. - an example: the dashboard, assembling of several small plugins...

community based development, for gateways, plugins, etc. plugin store

Incentives That it is a platform !

fully plugin based architecture async/exploits caching/updates (dynamic interface) authentication information independence of plugins / synchronization : pub/sub system readily available bricks

4 Discussion

Our target are communities that can install various personnalized instanced of the tool, adapted for their needs.

This proposition represents an ambitious challenge, but that seems

Big challenge: balancing heterogeneity, don't make something too much centralized.

* Propose a consistent interface Lower the barrier of entry for users present a unique semantic, a unique consistent interface, and ways to do things. integrae everything into a well recognized process = experimental lifecycle. Right balance of integration, not to converge to a monolithic architecture. Integration without losing information.

* But we propose a platform on top of which to build app, and social ties. Critical mass + becomes more simple to build on top of it. build a platform instead of a product community driven model + free software + consensus = effective Supporting other needed uses... anybody free to deploy, extend, learn, reuse

* ... and ensures sustainability. light framework with simple technologies: for maintainers, developers, etc. sustainability: thanks to the community model, needed developments done in an ad-hoc and distributed manner, ensuring the entry point to the federation remains as up to date as the resources provided (sustainability).

* Manage content and data cite the onelab portal Joomla: manage both content (natively) and data (plugins) could be plugged in with another framework Statistics, usage, etc. as different modules that can also be plugged into the main interface. Generating reports, etc.

* dissemination and visibility people won't lose their image easier for them to get visible (even small ones)

adding social ties

* what makes it work : semantic integration, simplification, etc. this is possible at the edge because : - trust (simplify authentication) - community - less robustness is required, allows for centralization and centralization allows mutualization, processing, accounting, etc. = what is needed at the community level

Semantic is key

Acknowledgements

TODO

References

1. todo: todo. todo (todo)
2. todo: todo. todo (todo)
3. todo: todo. todo (todo)
4. todo: todo. todo (todo)

5. Bourgeau, T., Augé, J., Friedman, T.: Tophat: supporting experiments through measurement infrastructure federation. Proceedings of TridentCom'2010 (May 2010)