ICT-317756

TRILOGY2

Trilogy 2: Building the Liquid Net

Specific Targeted Research Project

FP7 ICT Objective 1.1 The Network of the Future

WP2 D2.3: Final Liquid Net Architecture

Due date of deliverable: 30 June 2014

Actual submission date: 15 January 2015

The consortium and PO agreed to postpone to 31 December 2014

Start date of project	1 January 2013
Duration	36 months
Lead contractor for this deliverable	Telefónica, Investigacióy y Desarrollo, S.A.U.
Version	v1.0 , 15 January 2015
Confidentiality status	“Public”

Abstract

Trilogy 1 successfully defined a set of mechanisms to provide optimal fault tolerant transport across an IP network by creating and exploiting multiple simultaneous path across the network. Trilogy 2 sets out to extend the results of Trilogy 1 to be cross-resource, cross-layer, and cross-provider. Cross-resource seeks a generalisation to all IT resources including processing and storage. Cross-layer seeks to generalise beyond the simple two layers of Trilogy 1 and include intermediate layers. Cross-provider seeks to provide solutions which can work across commercial boundaries. The architecture described in this document gives a framework which directly addresses the requirements of cross-resource and cross-layer and implicitly cross-provider. The architecture is based on an extension of functional block methodology to incorporate virtualisation and as a result includes features beyond current state of the art. It has been already been presented to and adopted by the ETSI NFV ISG for its work.

Target Audience

The ultimate target audience for this deliverable is the community of knowledge engineers who define the structure of ICT systems, and those who define the standards and frameworks that are necessary for these ICT systems to interwork across the industry. In addition, this deliverable is also targeted at a) the project participants to ensure the whole is understood to be greater than the parts and b) the project's scientific advisory board and reviewers to articulate the approach being taken across the project in order to elicit useful feedback and criticism.

Impressum

Full project title	TRILOGY2: Building the Liquid Net
Title of the workpackage	WP2 Tussle over Liquidity
Editor	Pedro A. Aranda, TID
Project Co-ordinator	Marcelo Bagnulo Braun, UC3M
Copyright notice	© 2015 Participants in project TRILOGY2

Executive Summary

The Trilogy 2 project follows on from the Framework 7 Trilogy project (2008-2011). The original Trilogy project (Trilogy 1) developed mechanisms to create and control 'liquidity' of bandwidth in the Internet. The aim of Trilogy 2 is to extend the results of Trilogy 1 to create and control liquidity of other resources, in particular storage and processing. We also aim to extend the applicability of the Trilogy 2 results beyond the Internet to mobile devices and more general operator infrastructure. This document describes the extended Trilogy 2 architecture It builds on the initial architecture described in Deliverable D2.1 and collects the insights we have gained during the second year of the project. We have corrected our initial approach of a monolithic architecture and are instead approaching it as a cyclic, open ended development. This is the same approach taken by the Internet Engineering Task Force (IETF), where a set of principles guide the design of the specifications it produces. As these become available, the set of principles is revisited and some are re-scoped in line the findings so far.

We have split our principles into three sets. The first are axiomatic principles that may appear to be self-evident, but which we include because we wish to highlight some subtle aspects that apply to the Trilogy 2 architecture. The second set is a more traditional list of guiding principles. The third set has been derived from our work on the Reference Architecture of the ETSI Network Function Virtualisation Industry Study Group (ETSI NFV ISG).

Axiomatic Principles

- Portable Applications Create Liquidity
- We Must Have Metrics for All Resource Types
- Transport, Processing and Storage are all Strongly Inter-Dependant
- The Starting Default is a Real-Time Parallel Concurrent System
- Everything has its Own Defined Lifecycle
- The Starting Default is a Decompositional Model

Guiding Principles

- Name Resolution as a Network Function
- Biderectional Application Interfaces

Our work on the ETSI NFV ISG Reference Architecture has resulted in the following NFV-specific guiding principles:

- Network Function Virtualisation (NFV) Usage and Capacity Parameters Will Be Included in Commercial Boundaries
- NFV Usage and Capacity Parameters Must Be Consistent Across Heterogeneous Resources
- NFV Orchestration Should Have a Version of the End to End Principle
- NFV Orchestration Should Be Co-recursively Decomposable

From the beginning of the project, we have observed that there is no general framework within which solutions covering all three liquidity dimensions can be developed. In particular, the general techniques of functional modelling and functional specification cannot describe the essential properties of liquidity. This observation was also apparent in the work of the ETSI NFV ISG.

This deliverable documents our advances towards a simple architecture which can accurately describe and specify functionality, including processing and storage. We endeavour to decouple functional uses from the resources that implement those functions, as we believe this is the right way to allow for liquidity in the implementation.

Our basic functional architecture builds on the architecture of the ETSI NFV ISG which served as a basis for Deliverable D2.1. They are complemented in several fundamental aspects. Firstly, we believe that the Internet will evolve to a federation of liquidity based domains, which will need high level orchestration mechanisms, in the same way as today's cloud infrastructures are interconnected and provide means to the users to use resources across different domains. Secondly, we have also worked on mechanisms that allow the end-point to communicate and cooperate constructively to build end-to-end communication paths. Thirdly, we have investigated the aspects of Virtual Network Functions (VNFs) not covered in the ETSI NFV ISG like ways to combine (chain) them to implement more complex functionality and ways to pool VNFs to increase reliability. We recognise that there are research questions regarding them that are out of the scope of the ETSI NFV ISG and are in the process of creating a new research group (RG) in the Internet Research Task Force (IRTF) devoted to NFVs.

List of Authors

Authors	Pedro A. Aranda, Diego López, Andy Reid, John Thomson, Bob Briscoe, Costin Raiciu, Olivier Bonaventure, Giacomo Bernini, Gino Carrozzo, Jon Crowcroft, Anil Madhavapeddy, Toby Moncaster, Marcelo Bagnulo
Participants	TID, BT, OnApp, UPB, UCL-BE, NXW, UCam, UC3M
Work-package	WP2 : Tussle over Liquidity
Security	PUBLIC (PU)
Nature	R
Version	v1.0
Total number of pages	74

Contents

List of Figures

List of Tables

1 Introduction

Trilogy 2 sets out with an objective which can be stated in relatively simple terms. The objective is the creation of a framework for the flexible and optimal assignment of general ICT applications to the phyiscal resources which support them (that is processing, storage, and network resources). This framework includes the overall architecture, key algorithms for the optimal assignment, and key mechanisms for making applications portable between different possible resources.

However, this apparent simplicity belies the complexity of this objective. So the realistic aim is for Trilogy 2 to make a significant contribution towards this objective, acknowledging that there are elements required to fully meet the objective which are beyond the scope and timescale of this project.

1.1 General

The conventional method of managing a project, even a large project, is to start by scoping the project, capturing the external user requirements (normally through use cases) and then to break down the problem into a number of separate problems with a minimal but well defined interactions between the component parts, normally called the end-to-end architecture. This allows each component problem to be tackled as in independent problem and the challenge associated with each component problem tends to be a great deal simpler than the challenge of the overall objective. On the basis of the end-to-end architecture, all the conponents, once developed, can then be integrated together. The completed end-to-end system can then be tested against the external user requirements established in the frst stage. This is the essence of systems engineering.

However, developing a completed solution to meet the full scope of the Trilogy 2 objective is well beyond what can be realised addressed by such a ‘closed-form’ project. As is illustrated in Figure 1.1, we have adopted a more open-ended processes which is more analogous to the process adopted by the IETF in its early days when faced with the challenges of developing the Internet. Rather than scoping the problem by a requirements capture process, the scope can be established by stating objectives. In this open-ended approach, the architecture also becomes quite different. As there is no expectation that the project will complete and deliver a closed system, a traditional box and line diagram is not needed in the same way. Indeed, the problem may well be sufficiently poorly understood as to make any attempt at such an end-to-end architecture wrong, misleading, and unhelpful.

The first step in the Trilogy 2 architecture is to define a set of principles. The principles capture what is reliably known for the project. In the example of the IETF, these included the famous principles like the end-to-end principle and the liberal receive, conservative transmit principle. These capture important consequences of the basic observation that the network would be unreliable and implementations may be inexact and may vary over time. The project then identifies any gaps which mean that a closed project is not realistic. Solutions to gaps need only demonstrate how how a gap can be resolved rather than provide a finished component of an end-to-end solution. In effect this provides the start point for a future project which might develop

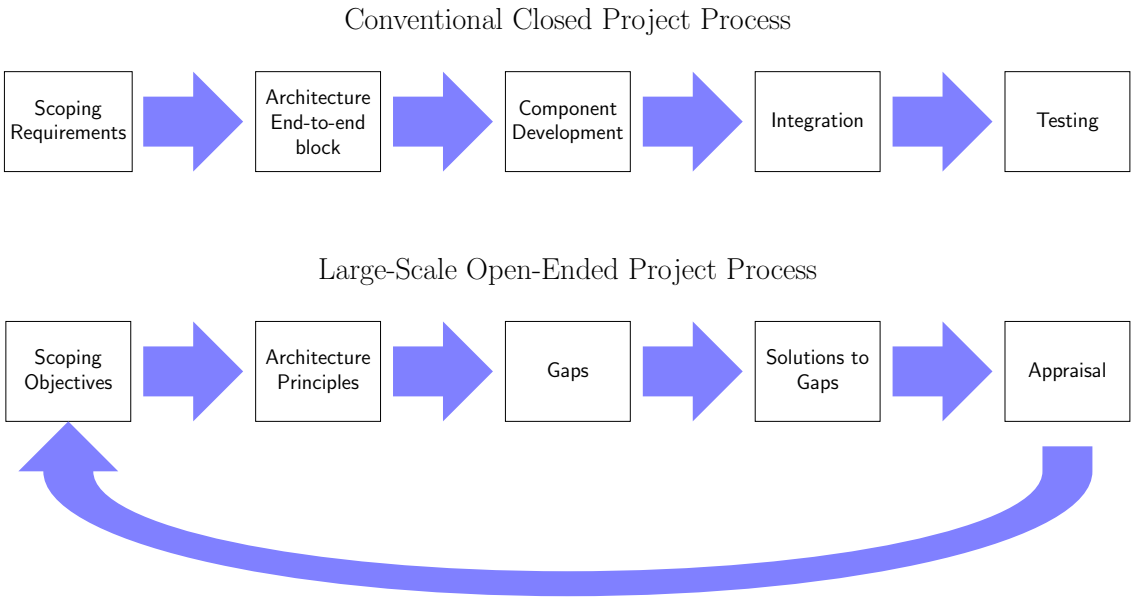

Figure 1.1: Comparison of One-Off Project with On-Going Project

such a component. For example, a solution to a gap might be an algorithm (rather than code implementing the algorithm in the context an end-to-end system), or is may be a development toolkit, or a domain specific language (rather than a specific solution using the toolkit or domain specific language).

1.2 Network Function Virtualisation

At the time of preparing this deliverable, the ETSI NFV ISG is finalising its first phase documentation and outlining a work programme for a second phase of activity. Broadly, the objective of the first phase was to establish and document the broad concept and architecture of NFV and also to demonstrate the key concepts with a series of Proof-of-concept (PoC) demonstrations. The scope of the first phase was explicitly defined as not specifying standards and the documents are formally classed as 'informative'. It is not intended that any implementation can be shown to be conformant to these documents.

This is addressed in the scope of the second phase where the primary objective is to achieve interoperability within the architecture. This means that the documentation of the second phase needs to be sufficient to decide the level of conformance needed for interoperability. However, the second phase is not limited to achieving interoperability within the architecture set out in the first phase. The scope will also include architectural aspects which were either not addressed in the first phase or unresolved in the first phase.

Indeed, the work of the first phase has revealed a number of significant gaps where the is currently no suitable technology available, standard or otherwise. It has become clear that the NFV standardisation project is a hybrid of the two development paths shown in Figure 1.1.

Where there is suitable technology and associated standards the project can proceed down the first path (the intention is that even within the second phase, the ETSI NFV ISG will only reference standards from other standards bodies and will not develop its own new specifications). The plan envisages that the ETSI NFV

ISG will further develop the archictecture such that components are specified to an interoperable level, and a new open source organisation, OpenNFV, will develop, integrate, and test these.

However, there are many aspects of NFV where this would be highly premature. These can follow the second path of establishing architectural principles, gap analysis, and development of solutions to the gaps.

Of particular note here is that one of the primary gaps is a suitable specification language to cover virtualisation which can allow a traditional 'box-and-line' architectural specification to be defined. Without this, a hybrid is approach is not practical this 'gap' is a fundamental requirement to progressing on the top path of Figure 1.1.

However, the methodology already set out in the interim architecture deliverable of Trilogy 2 is a solution to this critical gap and has already been taken up by the ETSI NFV ISG in the first phase. This methodology of unifying virtualisation and abstraction with functional block methodology is a key part of the architectural specification which will progress into the second phase.

The development of the methodology may be still at an early stage (see the gaps identified below) and their use within the ETSI NFV ISG may be even more rudimentary, but from this has come the key undertanding that the relationship between a VNF and its host NFV Infrastructure (NFVI) is that of an execution environment and it is emphatically not a functional interface of a functional block architecture.

This understanding, at least for the present, is allowing the second phase to include progress along the top path. However, the development of the methodology and it linkage to a converged ontology are a critical gap for NFV[1] and a major topic for continuing research.

1.3 Objectives

We have already noted that the overall objective can be simply stated—the creation of a framework for the flexible and optimal assignment of general ICT applications to the phyiscal resources which support them. However, we can also break down this single objective into a number of underlying objectives.

- Creating Liquidity
  - Architecture: to create a framework that allows the decoupling of applications from the resources used to implement along with mechanisms which create liquity of processing, storage, and network.
  - Standardisation: contribute to and seek agreement from appropriate standardisation bodies including IETF and ETSI NFV ISG.
  - Demonstrations of Concepts: to develop and be able to demonstrate solutions to particular areas in the liquidity architecture where existing solutions either do not exist or have performance issues.
- Controlling Liquidity

[1] This was also very evident from the discussions on NFV and Software Defined Network (SDN) at the stakeholder workshop for the future EU networks research programme (Sept 2014).

  - Architecture:
  - Standardisation
  - Demonstrations of Concepts

This requires a specification language(s) which can allow the aplications to be defined independent of their implementing resource but can be precisely executed on whichever resource is assigned to the application.

1.4 Document structure

The rest of this document is structured as follows:

- Chapter 2 describes the principles on which the Trilogy 2 architecture is founded,
- Chapter 3 describes the high level orchestration mechanisms that allow different domains in a network built using the Trilogy 2 architecture to communicate and cooperate
- Chapter 4 defines the new UNIs defined in the Trilogy 2 architecture,
- Chapter 5 describes the latest updates of the ETSI-NFV architecture and additional components that complement it and make the initial Trilogy 2 architecture the core of the final Trilogy 2 architecture, and
- Chapter 6 provides the conclusion and an outlook of the evolution of the project based on the presented architecture.

2 Principles

In this chapter, we introduce the principles that have guided the work of the project and identify gaps that will need to be covered in the final phase of the project and beyond. We split the principles between those that set out the foundations of the project, 'axiomatic principles', and the more conventional 'guiding principles'.

The set of axiomatic principles might be regarded as implicit. However, in the broad and challenging scope of Trilogy 2 we find it useful to set them out. In each case, the principle addresses an aspect of a scope or fundamental constraint which is not necessarily obvious. As an illustration, one of the axiomatic principles is that Trilogy 2 involves all of transport, processing and storage in a symmetric and tightly bound architecture. However, each of these have different architectural methodologies and information ontologies with subtle incompatibilities. By stating the axiomatic principle, we acknowledge the need to resolve the incompatibilities.

The set of guiding principles give architectural guidance to the project. Each guiding principle gives a clear direction for the form of solution the project is developing. This enables the different developments within the project to proceed with a basic understanding of the form of solutions being developed elsewhere.

In the sections below, we list the principles of our architectural work and document the reasoning behind them. Each principle is highlighted in a grey box.

2.1 Axiomatic Architectural Principles

Portable Applications Create Liquidity

This principle follows the observation that liquidity of processing loads between processing resources is considerably more complex than shifting the transport of a bit stream between different routes across a network. The key objective of Trilogy 2 is the creation of liquidity between ICT applications and the resources needed to realise the applications. While it is undoubtedly useful to have flexibility between a given application and a specific resource, the general benefits will only arise when a wide variety of applications can be supported on a wide variety of resources. This general benefit requires that applications be portable across many different resources.

Portability is therefore a central principle of liquidity. This can also be seen by analogy to markets. A commodity is normally said to be liquid if it can be readily sold at the prevailing market price without needing to be concerned about who the buyer might be. The liquid commodity is portable between buyers.

Portability is also an implicit feature of the Trilogy 1 architecture even if the portability is so well developed that it is hidden. We take for granted that data transport is portable between different routes across the network. This portability is achieved because the data requiring transport is reduced to a sequence of binary bits and the capacity of the resources is reduced to a capacity in binary bits per sec. The liquidity of transport

fundamentally depends on this reduction of data to bits and resource to bits per sec. However, portability of general ICT applications is much more complex because it isn't always so easy to define such clear parameters to measure liquidity (see the Principle on Metrics).

The portability of processing is a key principle for Trilogy 2. There are already many practical solutions to processing portability based on virtualisation with important examples being the Java run-time environment and the virtual machines of cloud computing. While these examples are directly usable in Trilogy 2 , they all tend to be tailored to particular types of applications - it is liquidity with a lot of viscosity. The principle for Trilogy 2 is an environment where the portability of processing is as liquid as the portability of transport and storage.

We Must Have Metrics for All Resource Types

Resource control is a fundamental requirement for resource liquidity. In turn this means resources must be able to be measured in such a fashion as to allow for accurate control. Transport capacity is measured in bits and bits per sec. These parameters are both profound and precise, arising from the mathematical foundations of information theory from which the 'bit' is derived. Traffic control mechanisms such as those that were fundamental to Trilogy 1 generally rely on measuring and controlling bit-rate. There may be secondary requirements needing control such as latency[1] but often these are related to bit-rate (latency is a combination of the fundamental time to send a given amount of data over a given distance coupled with the time to process it at any network elements and any time it spends in a queue).

On the face of it the number of bits should also be the necessary and sufficient parameter for storage. At a basic level this is true—the quantity of information to be stored and the capacity of a given storage medium are both measured in bits. However, latency is normally much more of a consideration as there is not the same fundamental minimum set by the laws of physics as there is with transport, and the variation in latency is affected by complex factors such as where the data actually resides on the physical medium. By storing information sufficiently close to where it is needed, latency can be made arbitrarily small. This is seen directly in the design of storage in data processing systems: registers, L1 cache, L2 cache, L3 cache, RAM (now often in a NUMA architecture), SSD and spinning disks form a clear hierarchy showing the trade-off between latency and storage capacity.

Processing however has no such fundamental parameter equivalent to the bit by which load and capacity can be measured. Typically for measurements of processing we can observe that:

- parameters tend to be informal and inaccurate, for example, processor clock speed and clock cycles are

[1] Other parameters include 'shared risk groups' susceptible to common mode failure. Historically, error rate would be naturally included as a parameter of the resource and this was indeed the focus of Shannon's seminal work. However, the direct result of that work is that errors can be and generally are removed from any meaningful consideration, for example, by management of optical power budgets and forward error correction schemes on optical transmission systems. The capacity of the system is effectively given as the error free capacity.

often used. However this is only a very loose indicator of the capacity required by different applications.

- parameters tend to be application specific, for example, there are numerous performance benchmarks but many different benchmark are needed in order to cover the diversity of applications.

- parameters depend on the processor architecture, for example, all of the above parameters will depend on the instruction set architecture of the processor.

Clearly one of the key things we need to consider in Trilogy 2 is this difficulty in measuring processing capacity.

Transport, Processing and Storage are all Strongly Inter-Dependent

When looking at any of the three fundamental resource types (transport, processing and storage), the focus is generally on that function and the other two are ignored. In Trilogy 2 we take as a starting point that all three are actually strongly linked.

As an example, storage requires a means of requesting writes to storage and requesting reads from the storage. Managing these read and write requests requires processing. Moreover, the location of the entity making these requests will be separated from the storage itself (by whatever distance) and therefore they require transport. So we can see that while storage is the primary function, strictly it also contains elements of processing and transport.

In fact, the same is true for the other elements of processing and transport. Processing is not really possible without storage and transport and transport is not really possible without processing and storage. From this it follows that it is sensible to consider each of the three elements as a combination of all three.

The Starting Default is a Real-Time Parallel Concurrent System

If Trilogy 2 applications are to be able to exploit parallel concurrent resources, then applications need to be specified in a way that exposes as much concurrency as possible. In addition, we observe that for real-time systems, speed of execution matters.

Instead of starting with sequential and regarding ‘real time’ as an extension of sequential, we regard temporal as the fundamental mode and sequential as a special case of temporal where time can be used to define sequence. This ensures the that starting point is universal (at least in the context of ICT systems) which is the objective of Trilogy 2. If cases do arise which are essentially offline and for which real time is not relevant, for example running simulation models, offline calculation of optimisation parameters, etc., then these can easily be regarded as special cases of a temporal system. In essence, we regard sequential and/or non real time systems as a subset of real systems.

Likewise we regard parallel as the fundamental construct and again sequential is a special redundant case of parallel.

Finally, we consider all systems as continuously executing. This is in contrast to a 'run to completion' model. This principle reflects the fact that Trilogy 2 is about operational real time systems and there are some possible 'hang-overs' from a computational model of processing based on solving maths problems. A summary might be that while the origins of computing was 'to do something in order to solve a maths problem', in Trilogy 2 we 'solve a maths problem in order to do something'.

Two important points arise from this recognition:

- Maths problems are not temporal in that the the time taken to solve the problem has no bearing on the truth of the answer, by contrast Trilogy 2 systems fundamentally exist in real time;
- One of the principle results of automata theory—that any parallel system has an equivalent sequential system—is important only in a non-spatial, non-temporal context. But when spatial distribution and time are included, systems may not be equivalent and the distributed system may well not be able to perform an operation in the same time that a non-distributed system can.

Everything has Its Own Defined Lifecycle

This principle reflects a more subtle observation but one which is central to Trilogy 2. We observe that a basic axiom of systems theory is that some things are static to the system and some things are dynamic. The things that are static are assumed to have always existed, but this is clearly not the case and they must have been created at some point in time. In the context of Trilogy 2, this becomes important as liquidity requires processes which automate the creation of things which are normally regarded as static. In other words, we observe that things that are static from the perspective of the operational system are also dynamic from the view of the liquidity control processes. We observe that whether something is static or dynamic is a matter of perspective.

The distinction between 'static' and 'dynamic' is especially important in the context of formal systems. In this context, the distinction is, at least in principle, mathematically defined and is precise. 'Dynamic' is expressed in the changing value of state variables while 'static' is expressed in the fixed and invariant transfer function of the system.

While this appears simple and clean, it does not account for the configuration of a system and in Trilogy 2 there is a considerable amount of configuration. Is the configuration static or dynamic? Or equivalently, is configuration a configuration of state or a configuration of a transfer function?

The principle we apply to resolve this question is that configuration is both, but over different timeshares. In a configuration time-frame, configuration is dynamic and is setting and changing state variables. However, once the configuration is made (and in a timeframe for which the configuration is held constant), the configuration

is static and hence the configuration state which is invariant in this timeframe is regarded as part of the transfer function. This model is fully recursive and so configuration may be layered.
This principle that everything is dynamic in its own time is a central concept in the unifying functional Trilogy 2 architecture.

The Starting Default is a Decompositional Model

This principle follows from the observation that there are dangers with following a systems architecture which relies on building complex systems from atomic components.
The ability to compose larger systems by treating smaller elements as components is a basic principle of large scale system design. While there are methodologies which encourage 'top-down' decomposition of systems, this tends to be more form the perspective of requirements capture rather than formal system specification. While it is a generalisation and far from universally true, the formal specification and design of systems is often compositional and constructional rather than decompositional and deconstructional. This follows from a 'Lego brick' approach where there is an assumption that there a number of basic building blocks - the Lego bricks - from which any required systems can be constructed.
There is a side affect of this type of approach in that components are regarded as being atomic, that is they are regarded as being fundamental and essential unchangeable and indivisible. This is a necessary part of the compositional and constructional approach; it must start from the construction with some fundamental components.
However, whatever these 'Lego bricks' are deemed to be, they are not fundamental or indivisible or unchangeable. These components are inevitably the product of some design process and they are only fundamental from the subjective perspective of the design process that was treating them as fundamental. For example, as highlighted in the 'everything is dynamic in its own timeframe' principle, the notion of unchangeable should always be regarded as relative and not absolute.
The Trilogy 2 principle is treat systems as decomposable and capable of deconstruction. The decompositive model does not require an atomic start point and so inherently includes the relative rather than the absolute nature of system components. This principle has its counterpart in mathematical specification which would now be called 'co-recursive' rather than 'recursive'. This also potentially follows through to methods of formal conformance verification which use mathematical co-inductive proof techniques rather than inductive proof.

2.2 Guiding Principles

Name Resolution as a Network Function

One of the legacies of the early design of the Internet is the use of addresses as both end-point identifiers and location pointers. This ambiguity leads to real issues with multi-homed end-points (one name, many addresses), multipath networks (one name, multiple viable paths) and workload/VM migration (one name, changing addresses).

For the liquid network we need clear separation between these, with names used as the primary mechanism to establish a communication flow between two endpoints, and the network (via NFV or middle-boxes) responsible to mediating the establishment of specific flows.

The separation on the naming scheme from the specifics of address resolution also informs how applications are architected. The exact capacity of a specific resource may not be known at the time an application is developed and the capacity may change over time. Therefore there is a basic requirement that applications are designed in such a way that they are free to utilise any available application and resource units. This means that applications should be designed in such a way that any one logical (name) component can be implemented across many parallel units of resource (addresses). This can also make the logical component tolerant to failures in the resource units.

This approach to the design of applications becomes a basic principle of their design and hence of all applications supported by Trilogy 2 liquidity.

Bidirectional Application Interfaces

The network needs bidirectional Application Programming Interface (API) coupled with improved information visibility. This allows the network to apply back-pressure to applications using more subtle mechanisms than the packet drop mechanism adopted by TCP.

Trilogy 1 came up with principle of Information Exposure [1]. That principle was about end points exposing congestion information to the network. We now build on that principle to suggest that in the liquid network this can be coupled with explicit API mechanisms to allow the network to apply direct back-pressure to applications. A similar trend has been observed with Backward Congestion Notification for Ethernet.

A key implication of this is that the network can now build in a notion of explicit scheduling, since it has a means to communicate with applications that have already established a flow. It is essential for the network to support such scheduling across all the layers that Trilogy 2 is concerned with, including compute resources (virtual machines), network bandwidth and storage.

However, it is important to note that the scope of Trilogy 2 is sufficiently big and there are sufficient existing,

non-liquid solutions to the same problems, that creating a solution which requires completely new application development to exploit Trilogy 2 is unlikely to commercially viable. This general principle is that the way Trilogy 2 supports applications should be evolutionary. Wherever possible existing applications should be able to exploit Trilogy 2 with minimal redevelopment and, more generally, it should be possible to introduce Trilogy 2 in incremental steps where every step is independently commercial viable but where the incremental benefits are cumulative[2].

As a result of out work on the NFV architecture proposed by the ETSI NFV ISG, we have also observed that there may be a commercial boundary between NFV Infrastructure and Virtual Network Function hosted on the Infrastructure, which needs to be taken care of. This observation has led to the following principles for Trilogy 2 .

NFV Usage and Capacity Parameters Will Be Included in Commercial Boundaries

One of the defining characteristics of NFV is the separation of the software which defines the network functions from the hardware which implements the network functions. This separation allows for the creation of liquidity and its control inherent in NFV.

While in the context of a single operator there considerable benefit of decoupling the supply and operations of the hardware and the software, there is much greater scope for benefit if the hardware and the software can be owned and operated by different operators. In this case the NFVI can be offered as a service - NFVIaaS - and this is a model being actively developed by the ETSI NFV ISG.

As and when the NFVI is offered as a service, this means that on the supply side the resources of the the NFVI and their capacity will need to be parameterised as part of the service. On the demand side, the usage requirements of the VNFs will also need to be parameterised in a way that matches the resource capacity parameterisation. Moreover, these parameterisations need to be sufficiently open, accurate, and robust to work in the context of the open commercial interface.

NFV Usage and Capacity Parameters Must Be Consistent Across Heterogeneous Resources

In its fundamental conception, the NFVI is based on generic hardware, specifically, generic servers, generic storage, and generic switches (for example SDN controlled switches). However, in practice 'generic' is more of a commercial definition than a technical specification and there are important technical differences between resources which are 'generic' from a commercial point of view. There are also some apparently minor technical details which can have significant consequences for the capacity of the resources in the context of NFV (an example is the CPU cache architecture which can significantly impact the achievable

[2] An example of a major development where this principle was not applied and the introduction has been extremely problematic is IPv6.

packet throughput of a CPU core).

ETSI NFV ISG early on decided that to tie down and specify these details would be counter to its primary objective as the hardware would no longer be 'generic' but specifically tailored to the needs of NFV. Instead, there has been a clear decision that these details of the NFVI resources must be capable of be abstracted away and hidden from the software VNFs using the NFVI resources. It therefore follows that parameterisation of the resource capacity and the VNF usage is also capable of abstraction and independent of the details of the heterogeneous resources. In effect, there is consequential requirement from this principle that different detailed versions of the same generic type of resource, for example a server, can have a different values of capacity, but the capacity parameterisation must be the same.

NFV Orchestration Should Have a Version of the End to End Principle

In the end to end principle, certainly as articulated by the IETF, the ends do not presume to know about the internal state of the middle or to control the state of the middle. The middle is treated as an independent, unreliable black box. The result is a huge reduction in complexity as their is no coupling of the state of the ends with the state in the middle. This principle is also critical to the independent evolution of the ends and the middle such that when the end to end principle is maintained, new solutions can be developed in the middle without affecting existing solutions at he ends at vice versa. In addition, the end to end principle means that it is inherent and simple to have many different ends independently using the middle concurrently.

In the Internet, the end to end principle is applied in the resource allocation of the network capacity. The network in the middle is regarded as a independent unreliable black box. The end systems control the rate at which they load use age onto the network resources based solely on the externally observable reliability (or unreliability) of the black box network.

With NFV, the NFVI provides the resources and is the middle in the end to end principle. The NFV Orchestrator (NFVO) is the end system which places usages upon the resources. Following an end to end principle, the NFVO should not have knowledge of the resource state of the NFVI and vice versa: the NFVI should be an unreliable black box. A requirement on the NFVO which follows from this principle is that the NFVO needs to 'discover and request' resources, it must not 'know and assign' resources. The adoption of this principle makes possible the extension of TCP and MPTCP liquidity control algorithms to NFV. A further most important consequence for NFV of this principle is that an NFVI can be concurrently manage usage requests from multiple NFVOs. In should be noted that many early approaches to NFVO have not followed this principle.

However, under a heading of 'horizontal equality' of orchestration, this principle have been contributed to and accepted by ETSI NFV ISG [21].

NFV Orchestration Should Be Co-recursively Decomposable

The final guiding principle relating to NFV concerns the decomposition of orchestration. ETSI NFV ISG has already agreed that orchestration should comprise four independent layers of orchestration.

- orchestration of VNFs into network services (by the NFVO service layer)
- orchestration of VNF Components (VNFCs) into VNFs (by the VNF manager)
- orchestration of distributed, heterogeneous NFVI resources needed for VNFC instantiation and their interconnection (by the VNF Orchestration (VNFO) resource layer)
- orchestration of local NFVI resources (by the Virtual Infrastructure Manager (VIM))

This principle reinforces this layering and that the layering should create independence and autonomy of operations at each layer. However, this principle also asserts that the general lifecycle management and operations at each layers should be broadly the same. They should be corecursively similar. Note this is strictly co-recursion rather than recursion as it is decompositional rather than compostional. When a higher layer decomposes an overall operation and one component at that higher layer requires an operation of the layer beneath it, the overall operational at this lower layer has that same abstract form as the overall operation at the higher layer. As this is decomposition, there is no predefined bottom layer, and the decomposition process halts as a matter of practical convenience.
This principle has been presented to and accepted by ETSI NFV ISG as 'vertical equality' of orchestration [21].

3 High level orchestration of domains

3.1 Introduction

Orchestration is used to control and manage the workflows of complex computing systems and the associated services. In literature such as Erl [8] it is defined as a part of service orientated architectures (as WS-BPEL) and is used in conjunction with choreography (through WS-CDL).

> *With orchestration, different processes can be connected without having to redevelop the solutions that originally automated the processes individually. Orchestration bridges this gap by introducing new workflow logic. Further, the use of orchestration can significantly reduce the complexity of solution environments. Workflow logic is abstracted and more easily maintained than when embedded within individual solution components. – [8, Ch. 2]*

3.2 What is Orchestration

Orchestration is about coordination of a complex set of resources so that they can provide a given service or services without the need to introduce complex new logic. In order to effectively orchestrate a system the orchestration system has to have full observability of the domain it is managing and have sufficient information and control of the system in order to fulfil the specified changes.

- Events are points at which significant changes to the workflow occur. Events usually have a location and a time at which they occur as well as a duration.
- Time is an important axis on which events may occur in a workflow. Although it may be sufficient to have an ordering of events that list the detail of how operations are executed relative to each other, in other cases it may be used to synchronise with external events.
- User-actions are changes made to the system by users that affect the work-flow
- Deterministic behaviour of events means that the model is sufficiently detailed that given the same set of inputs the output would always be as expected.
- Non-deterministic behaviour cannot be repeated and occurs due to events out of control of the system.
- Functional approaches are needed for orchestration as they describe the intention and purpose of the process that will be performed. How this is achieved is implementation specific and may vary in different contexts.
- Service composition describes how the services are related to each other and how they interact.
- Provenance defines the history and ownership of the entity. How an entity came to be in its current state is due to its provenance.

- Roles and accountability are important aspects of orchestration as it may be necessary to delegate certain tasks to other services when performing orchestration. In order to log and trace the history of events the changes made by differet users and parts of the system should be captured.

- Service Orientated Architectures normally deal with the orchestration and in particular communication and protocols when multiple services are linked together.

Probably the best way to show the concept of orchestratin is with the use of a trivial example.
An e-commerce service exists on provider site *IA*. If the resources provided by Infrastructure provider *IB* are cheaper or there are other business requirements that entail migrating resources then to migrate the e-commerce service requires the migration of the underlying components and reconfiguration on the remote provider. The e-commerce service comprises a web-server and a database. If the web-server and database are contained in a single VM then the service is fully comprised in one container that can be shifted as a logical unit. In the case of large enterprise applications and more complex examples, these will likely be separated as individual services that are then connected via APIs and established protocols. To migrate the e-commerce service, firstly the web-service has to be shut down (there may be cases where the web-service cannot be unavailable that would extend this scenario using load-balancers). Once the web-service has been shut down, the database would need to be synchronised to ensure that all the data is in a synchronised state. The Virtual Machines containing the web-service and the database service would then be migrated (the order is not important in this example). Once on the remote provider, *IB*, the database service would need to be started before the web-service. There will be changes to the configuration, given that the public IP addresses and certain other properties of the system will be different after the migration. After the services have been reconfigured there may need to be further actions such as the set up of associated logging tools, acquiring a new DNS entry and interacting with a remote service provider to notify of the change of global IP address. Orchestration should take the steps to ensure that the configuration of the services is well established and also provide a set of steps to follow in the case of any failure.
Already in this example there are the notions of a context environment (where the service resides when executing), ordering of services, dependency graphs and state configurations. More complicated examples will extend this to take into consideration more specific timing constraints, failure handling, context-environment mismatch resolution and other properties to make the orchestration more robust.
The idea of provisioning, deploying and configuring resources to make them interoperate on different cloud infrastructures is not new and has been investigated by many groups as highlighted later in Table 3.1. In 2012, Dr. Angel Diaz summarised these ideas that were being worked on in TOSCA into a set of conceptual diagrams in a blog post[1]. One conceptual diagram that highlights these phases, the actors involved and services running over physical resources in Orchestration can be seen in Figure 3.8.

[1] http://thoughtsoncloud.com/2012/01/tosca-and-cloud-services-orchestration-compose-once-play-on-any-cloud/

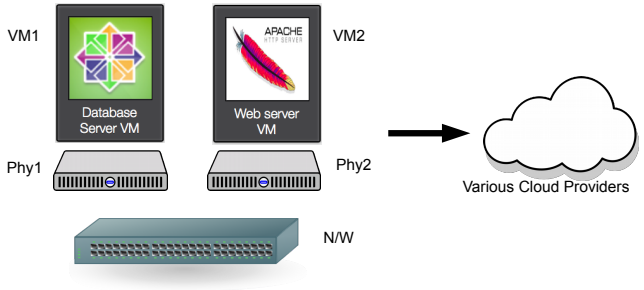

Figure 3.1: Services exist that are to be migrated and set up using an orchestration engine

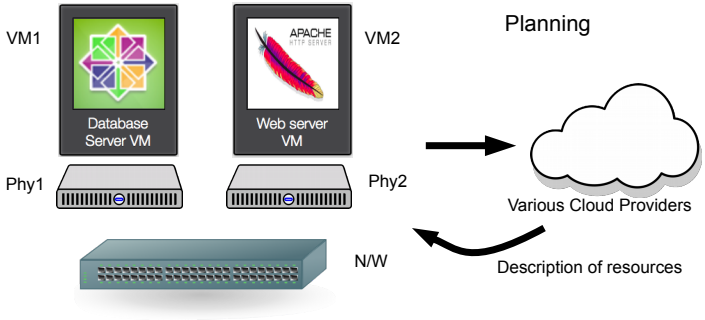

Figure 3.2: Use the Information model to describe the resources and the requirements

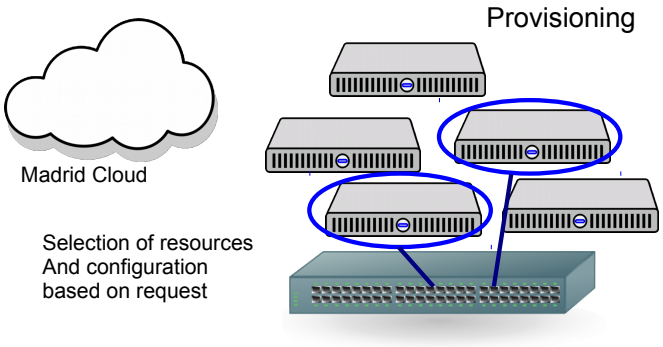

Figure 3.3: Orchestration system selects the resources based on the requirements and capabilities

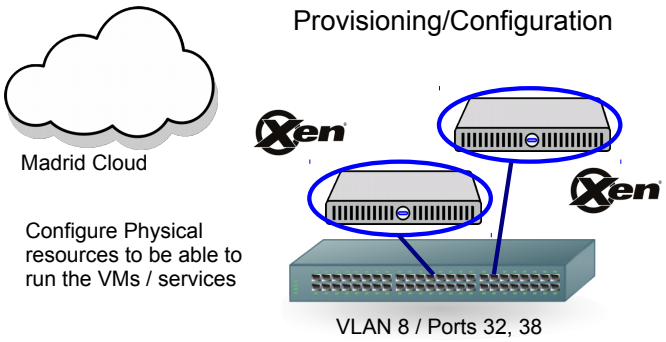

Figure 3.4: The Orchestration system configures the network and physical infrastructure first

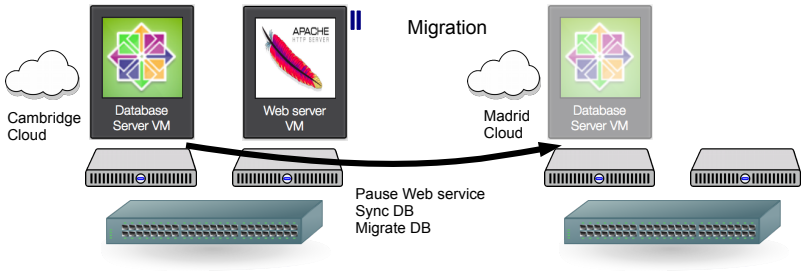

Figure 3.5: The relevant services are migrated. The Orchestration engine handles the ordering and timing

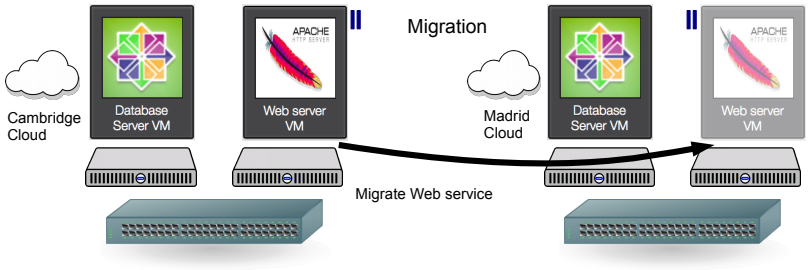

Figure 3.6: Once all the services have reached a set point, the Orchestration system can then continue with the plan

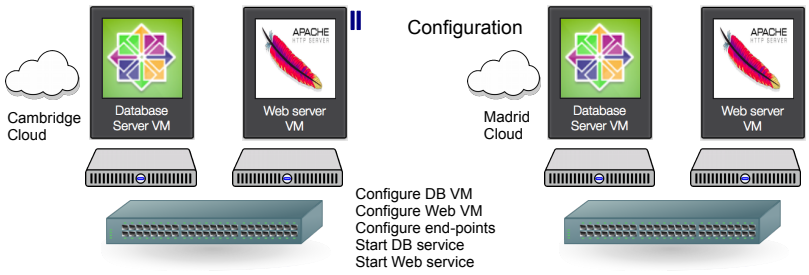

Figure 3.7: The Orchestration system then handles the reconfiguration of external sites to use the modified resources and handles the cleanup of the original system

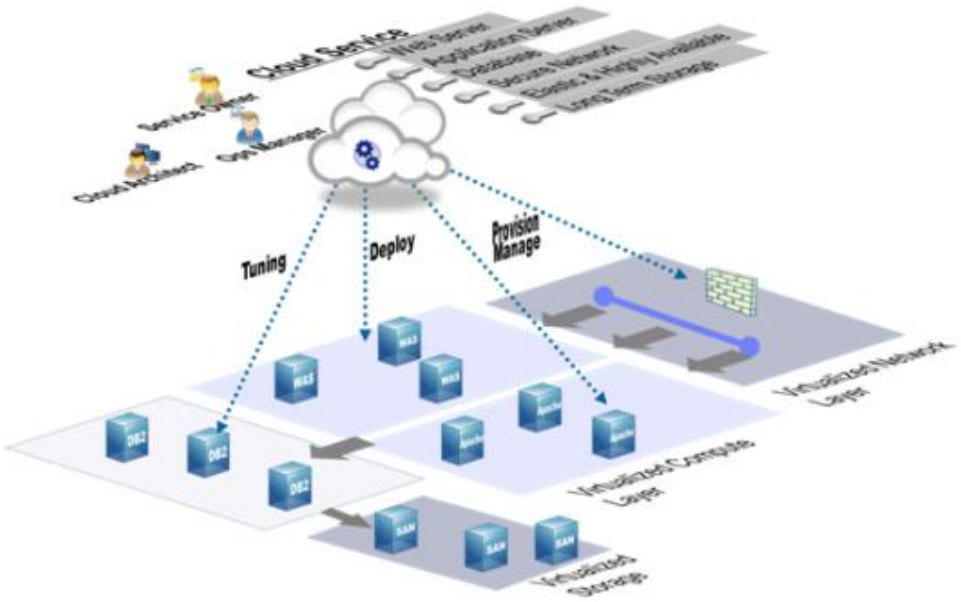

Figure 3.8: Conceptual diagram showing the phases, actors and services involved in Orchestration

3.3 How orchestration is used in Trilogy 2

In the context of Trilogy 2 we use orchestration as described by the combination of SOA orchestration and choreography terms, also known as Cloud Orchestration. We use orchestration to describe how resources should be provisioned to meet the requirements of the various use-cases described in WP3.

The way in which resources are provisioned is an important part of the workflow for making resources available and has to be addressed to enable resource liquidity across platforms. As such it is an important part of the architecture that has to be well defined.

Current best of breed orchestration systems are starting to use monitoring tools but for Trilogy 2 we will take this as being a more general testing and monitoring system that will need to interact throughout the orchestration process for the entire life-cycle of the services.

A generic orchestration system will use the available APIs to control different Cloud platforms but will also need to analyse the features of the API to reason and understand whether certain processes are available given the API implementations available. For instance if orchestration of an OpenStack implementation did not report when the web-service had been shut down, then it requires non-deterministic behaviour to arbitrarily decide when and whether to power down the database service.

3.4 Orchestration Scope

In order to effectively orchestrate the elements in the platform it is important that the entire state transition of the workflow is known and that the system behaves in a deterministic manner. What makes the process more complicated is that the hardware and configuration that the workloads are run on may not be known prior to the migration of resources and this is definitely the case when multiple operators are used (see Section 3.4.2).

3.4.1 Orchestration in the local domain

Orchestration in the local domain refers to orchestration within the control of a single enterprise. There may be multiple independent sub-entities within the enterprise that restrict or enforce certain policies on the use of resources but with sufficient effort, changes can be incorporated into an enterprise. There may be many layers of business processes and operational workflows that are automated or in some cases require manual intervention or decisions. Orchestration should reconfigure and modify the resources in such a way to maintain the attributes specified in any contractual obligations and/or SLAs. This might for example mean that an online service must ensure thar a user can perform any of a set of defined operations, such as read web-pages, perform purchases or modify databases within certain acceptable latency time-frames for at least 99.99% of the time. An overall orchestration framework may have to balance the business requirements from many areas and make trade-offs between incompatible service operations and then this becomes a form of optimisation problem with variables that may take ranges of values. To continue the web-service example, it may be that maintaining 99.99% availability is important but that the cost to meet this is too expensive, or the resources are out of control of the operator in which case the constraints may have to be relaxed.

A model of the local resources and process capabilities has to be generated to answer the queries that the orchestration reasoning system will use to determine whether operations are possible and in what way they should be performed. The orchestration reasoner will then have to determine the mapping of resources and how to perform the operations. Some queries can be run as background tasks for workloads that do not need fast responses but for a set of user operations the responsiveness of the system will also provide a constraint for determining whether a solution can be found in an acceptable time. The more queries that are generated and the greater the complexity of the problems, the more compute, storage and memory resources the orchestration system will be needed and this can be viewed as an overhead that moves resources away from other potential uses.

3.4.2 Orchestration between multiple domains

Orchestration that works across domains requires a knowledge of what can be performed on the other system. In SOA models it is known as choreography as it is the cross boundary form of managing services. It has the same constraints as the separate logical units in a local domain but additionaly has certain contractual obligations that need to be met and cross domain business constraints. Users that may be authorised in the local domain have to gain trust and access to key functionality in the other domain. Typically locally authorised representatives will communicate with someone of the right area in the other domain and then once communication is established ask for remote procedures to be run on their behalf. A Federated system allows for each system to be owned and managed by different entities but have some higher level goals and objectives that are mutually beneficial that can be met through the same or different processes run by the independent enterprises.

3.5 Efforts from other groups

There are many efforts from different organisations and research groups that have investigated orchestration from many different perspectives. Service Orientated Architecture (SOA) type of approaches define the way in which business services should interact with each other.

For SOA there are a set of protocols and approaches that describe services and how they shoud interact. Normally these approaches look at detailing a single protocol or communication channel and do not describe how services should interact at a higher level. Some examples of these types of approaches include: Web services orchestration [5], SOAP, WSDL, UDDI and WS-Security. There are also a number of efforts that have been discontinued that include: WSFL, BPML, WSCL and WSCI, BPEL4WS and WS-CDL.

3.5.1 Network management and orchestration

Trilogy 2 attempts to work with resources at multiple different levels and across different domains. As such there is a requirement to have an orchestration system that is flexible and extensible to work with the different resource types and services available. To indicate the variety and breadth of the different efforts within IaaS and PaaS platforms an attempt has been made to collate a non-exhaustive set into Table 3.1.

Table 3.1: Orchestration efforts by different groups

Entity/group	Orchestration System	Site / more information
IEEE[2]	P2302 - Intercloud	http://standards.ieee.org/develop/project/2302.html
ETSI	NFV-MANO	http://network-functions-virtualization.com/mano.html
Amazon	AWS CloudFormation	http://aws.amazon.com/cloudformation/details/
Google	Kubernetes	https://cloud.google.com/compute/docs/containers
OASIS	TOSCA	https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
ISO	JTC 1/SC 38 WG3	http://www.iso.org/iso/jtc1_sc38_home
OpenStack	HEAT	https://wiki.openstack.org/wiki/Heat
Amazon	AWS CloudFormation	http://aws.amazon.com/cloudformation/
Rackspace	Cloud Orchestration	http://docs.rackspace.com/orchestration/api/v1/orchestration-devguide/content/overview.html
HP	Operations Orchestration (HP OO)	http://www8.hp.com/uk/en/software-solutions/operations-orchestration-it-process-automation/
ServiceNow	ServiceNow Orchestration	http://www.servicenow.com/products/orchestration.html

Continued on next page

[2] http://grouper.ieee.org/groups/2302/

Table 3.1 – *Continued from previous page*

Entity/group	**Orchestration System**	**Site / more information**
Ubuntu	Juju	http://www.ubuntu.com/cloud/tools/juju
Cloudify	Cloudify 3.0	http://getcloudify.org/cloud_orchestration_cloud_automation.html
VMWare	vRealize Orchestrator	http://www.vmware.com/products/vrealize-orchestrator
SAP	SAP Process Orchestration	http://scn.sap.com/community/developer-center/process-orchestration
Docker	Machine / Swarm / Compose (Fig)	http://blog.docker.com/2014/12/docker-announces-orchestration-for-multi-container-distributed-apps/
Citrix (/CloudStack)	CloudPlatform	http://www.citrix.com/products/cloudplatform/overview.html
Ansible	Playbooks + Inventory	http://docs.ansible.com/playbooks.html
SaltStack	Salt	https://www.saltstack.com/community/
GoGrid	OpenOrchestration	www.openorchestration.org
Cisco	Director / Intelligent Automation for Cloud / Process Orchestrator	http://www.cisco.com/c/en/us/products/cloud-systems-management/intelligent-automation-cloud/index.html
Dell	CloudManager	http://software.dell.com/products/cloud-manager/

Continued on next page

Table 3.1 – *Continued from previous page*

Entity/group	**Orchestration System**	**Site / more information**
Puppet	Marionette Collective	http://puppetlabs.com/mcollective
OpsCode	Chef Cluster Orchestration	https://wiki.opscode.com/display/chef/Cluster+Orchestration
CFEngine	CFEngine	https://auth.cfengine.com/archive/manuals/st-orchestrate#How-does-CFEngine-deal-with-modularity-and-orchestration_003f
IBM	Cloud Orchestrator	http://www-03.ibm.com/software/products/en/ibm-cloud-orchestrator
Abiquo	anyCloud Cloud Orchestration	http://www.abiquo.com/anycloud/

3.5.2 IEEE Intercloud Project

One standardisation effort that is of particular interest for the Trilogy 2 architecture is that of the InterCloud Project. There have been a set of efforts that have consolidated a number of outputs from different EC Projects and have matured into a set of IEEE Working Groups and an associated umbrella project, the IEEE Intercloud Project. It originates from several collaborative efforts that have seen the need for a converged system and have predicted the growth in inter-cloud communications. Of particular notes are the standards group, IEEE P2302 Standards Work and the draft standard IEEE P2302/D0.2, Draft Standard for Intercloud Interoperability and Federation (SIIF). InterCloud has the backing of a mixture of industry, service providers and operators including Orange, 6fusion, Cloudscaling, Telxx Group, Juniper Networks and DOCOMO Innovations and also had 21 founding members of the interoperability testbed[3]. The IEEE Intercloud Testbed Project benefits from resource description as worked on within the mOSAIC[4] EC-FP7 Project. The working groups from InterCloud have also been cited as being possible standards for Cloud Standardisation[17]. One industry analyst, John Messina, a senior member with the National Institute of Standards and Technology's cloud computing program, speaking at a conference in May 2013[5], predicted that five years from now a suite of international interoperability standards will lead to a cloud of clouds, or "intercloud," where there will be tight integration between multiple clouds. Efforts like that of Intercloud are becoming more dominant and it will take large industrial collaboration, standardisation or a movement away from the de-facto Amazon cloud services to cause shifts in alignments to a set of universally adopted approaches. What is not clear is if like in the case of other standards battles such as HD-DVD and Blu-ray whether the groups involved will seek to find a common solution or if there will be a continuation of multiple, incompatible standards.

The InterCloud project suggests using a set of communication protocols that already exist for communicating between various elements in the architecture. As can be seen in Figure 3.9 they have suggestions for the communications protocols. These include:

- XMPP for control plane and sending messages between clouds. In particular they suggest that XMPP is useful for lightweight services that require asynchronous behaviour. In particular they use a set of XMPP extensions - XEP-0244 IO Data[6].
- For the services framework they suggest using Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST)
- Security services will be encapsulated using TLS then SASL
- Service invocation using xws4j - XEP, XMPP Web Services for Java
- SAML for identity and authentication

[3] http://www.businesscloudnews.com/2013/10/11/ieee-forms-cloud-interoperability-testbed/

[4] http://www.mosaic-cloud.eu/

[5] http://gcn.com/articles/2013/05/31/cloud-of-clouds-5-years-in-future.aspx

[6] http://xmpp.org/extensions/xep-0244.html

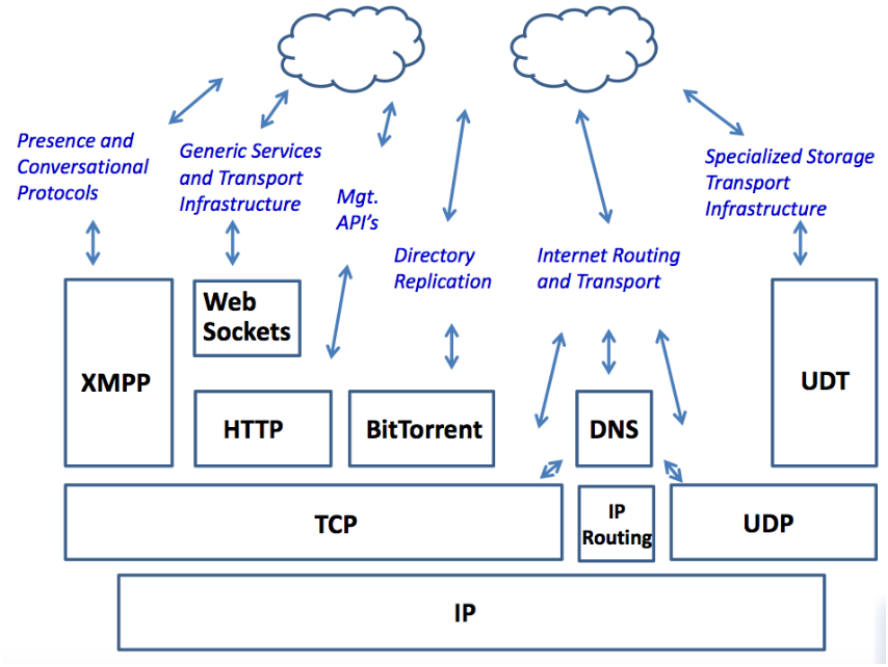

Figure 3.9: InterCloud suggested protocols for communication beteween the different elements.

There are also other promising services including WS- (Web Service) -Federation and Liberty ID-FF that is now part of Kantara Initiative [14]. XMPP using a federated model has also been investigated.

The elements that have been identified as being important for Trilogy 2 architecture are the inter-communications language, the Information Model and also the orchestration of multiple services. In Trilogy 2 the Orchestration elements therefore focus on the InterCloud 'Exchange Service Discovery and Communication' architectural blocks.

4 Bringing control back to the endpoints

A key aim of the Trilogy 2 project is to ensure that existing in-network functionality, currently set in hardware and difficult to upgrade or scale, can be run as software on commodity platforms with a view to accomodating the stringent scaling requirements of in-network processing. The strong push towards NFV will thus make it possible to run more scalable middleboxes, able to start up when they are needed, empowering operators to quickly adapt and optimize their networks in response to changing customer traffic requirements.

This will lead to a proliferation of middlebox deployments that is very appealing to network operators and downright scary to some of the purists of the Internet, which view the end-to-end principle as set in stone. A read over the recent mailing list discussions in the TCP Maintenance and Minor Modifications Working Group at the IETF regarding the extension of the TCP option space will quickly show the diverging attitudes towards middleboxes ranging from resignation to dismay.

Clearly, there is a tussle [7] between the endpoints requirements for unlimited, un-mediated connectivity and the need of network operators to manage their network properly. The job of the Trilogy 2 architecture is not to decide the potential winners of this tussle, but to provide the right mechanism such that the different parties can collaborate efficiently.

To this end, in this section we analyze the Internet from the point of view of the end-clients: mobile phones, desktops and servers. To what extent is the Internet liquid from this viewpoint: is it possible to utilize resources where they exist? Middleboxes such as NATs restrict inbound connectivity, firewalls limit allowed traffic to a few ports, and app-specific optimizations by operators might not be wanted by the applications in the first place. In fact, end systems have little control over the service they get from the network and tend to be conservative when choosing transport protocols: an app that prefers UDP will typically just use TCP for fear of getting blocked by the network, losing out in efficiency. This results in a rigid Internet where resources are underutilized, and end-system liquidity is rather limited. By optimizing their network to suit what they think apps want, network operators are effectively pushing many novel apps to utilize inneficient transports, which require even more optimization, leading to a vicious circle.

We begin our discussion by looking in detail at the root problem affecting endpoints, the contract offered by the network, and then outline a few approaches pursued in Trilogy 2 in which endpoints can wrestle back some control over the service they're receiving:

- Ninja tunneling offers an efficient way to choose the best protocol for getting through the network, nullifying the effect of app-optimizers.
- We discuss a constructive approach in which this tussle between endpoints and the network can be solved.
- Opportunistic Encryption forces middleboxes to act as an active man-in-the-middle if they want to snoop on and modify the TCP payload.

4.1 A Broken Contract

The original Internet architecture offered a clean contract to endpoints: packets sent will be delivered unmodified [1] or dropped when there is congestion. The proliferation of middleboxes has broken this simple contract to the point where the service the Internet provides to endpoints is entirely unpredictable:

- Reachability depends on fields in the packet header and even the payload, as firewalls strive to contain increasing levels of malicious traffic targeted at vulnerable endpoint software.
- Packets can be modified en-route by boxes that understand the higher level protocol (either TCP or app-level) and optimize it. For instance, NATs support FTP by rewriting the IP address of the sender inside the TCP payload to match the address of the NAT. As the NAT's IP address will likely have a different length, this forces NATs to also modify sequence and acknowledgment numbers. Other performance enhancing middleboxes are discussed in [6].

Firewalls not only drop all unknown protocols or extensions (e.g. SCTP [22], ECN [20]), but they also constrain reachability for traditional protocols: there is no guarantee that UDP or TCP outside ports 80/443 will get through many networks including office, cellular or hotspots [13].

This pushes most apps to rely on tunneling to reliably get through networks. HTTP is a favourite amongst mobile apps, and it has even been touted as the new hourglass of the Internet [19]. However, tunneling adds framing overhead and the effect is quite pronounced when the tunneled traffic is UDP-like (e.g. VoIP): in such cases (useless) retransmissions and head-of-line blocking increase jitter and degrade app-performance.

A minority of apps use adaptive tunneling to ensure reachability and the smallest possible overhead: for instance, Skype tries UDP, then TCP and finally HTTP or HTTPS. This approach is also suboptimal: certain middleboxes rate limit UDP tunnels to a level where Skype can check reachability but can't make calls [2].

Content-modifying middleboxes are more problematic: they optimize for known apps (e.g. FTP/HTTP) but can break apps that utilize the same port numbers as the known apps. For instance, HTTP parsers can reply with cached content instead of forwarding the request to the server which can break end-to-end semantics of apps tunneling over HTTP. Because of this, apps are forced to tunnel over HTTPS, thus hiding their traffic from the operator. This outcome is suboptimal for all parties: mobiles spend more energy to encrypt and decrypt traffic (15% in our tests on a Galaxy Nexus) and the operator can't see the traffic anymore and can't protect its customers and network against attacks.

Content modifying middleboxes also increase complexity in new protocols. Multipath TCP [9], a TCP extension that allows using multiple paths in a single TCP connection, includes a *redundant* checksum in the DSS option to ensure it can function correctly with content-changing middleboxes: when a change in payload is detected MPTCP either closes the affected subflow or reverts to plain TCP if the affected subflow is the only available one.

[1] With the exception of the TTL and checksum fields.
[2] Discussion with Romanian cellular operators.

To summarize, creating new apps is a difficult proposal because one has no idea what the network will do to one's packets. Consider an online gaming app: it can be conservative and use HTTPS to avoid packet modifications by the network and ensure reachability but this results in a highly inneficient use of network resources and poor user experience. The gaming app can be bold and use UDP targetting efficiency but this can lead to a lack of connectivity.

4.1.1 Steps towards a solution

What are the tools available to endpoints that can be used to gain control over the service they receive from the network? Encryption stands out, but even simple checksums will detect changes as long as in network boxes do not modify them (e.g. the MPTCP approach). Using HTTPS for encryption works but negotiation of a new protocol over HTTPS takes many RTTs and one still needs to add functionality over HTTPS to implement a demultiplexing layer that diverts traffic to the different apps—the equivalent of port numbers in transport protocols such as UDP or TCP.

It would be better if encryption was available at the transport layer so that all apps can use it naturally, without any changes: in Trilogy 2 we are pursuing the standardization of mechanisms for encryption of TCP streams, through the creation of the TCPINC IETF Working Group. The architectural design of such a solution is described in the next section.

Ubiquitous traffic encryption prevents packet modifications by making it very expensive for operators to modify or snoop on traffic: operators must perform an active man in the middle attack on every connection. Encryption does not guarantee efficiency, though: operators can still degrade performance by detecting and shaping encrypted traffic. This might be the case in the early stages of deployment, and it may push endpoints away from opportunistic encryption.

To liquidize the usage of the network, we propose: a) ninja tunneling, a solution where endpoints use a series of tunnels in parallel and stripe traffic over all of them; the trick is to not send too much traffic over inneficient tunnels, and we show this can be achieved by leveraging MPTCP congestion control, and b) show how Multipath TCP can be used to avoid IDS detection by ISPs.

4.2 Ninja tunnels: efficiently hiding network traffic

If network operators are unwilling to cooperate with endpoint apps, we propose that endpoints should simply force the network to allow reachability and avoid packet changes by using a technique we call *ninja tunneling*. With ninja tunneling, every endpoint always uses a set of tunnels to get through their first hop network operator, such as cellular, DSL or hotspot. The set of tunnels can dynamically change over time, and it can include any from the following non-exhaustive list: native IP, UDP, TCP, HTTP, HTTPS, DNS, covert channels, etc. The set of tunnels is forever changing, perhaps in response to network behaviour.

The key to ensure *ninja tunneling*'s success is that tunnels are invisible to applications: unmodified apps should benefit from it. The ninja tunnels will be deployed as software in the (mobile) operating system and at content/cloud providers that terminate the tunnels initiated by the clients. Should all tunnels originating

from a mobile user be terminated by the same cloud machine, or can we use tunnels terminated by different machines spread through the Internet? The latter is more attractive because it would make detection and filtering of ninja tunnels much more difficult.

The answer to the question above is closely tied with another issue: how should application traffic be spread over this dynamic collection of tunnels? In this document, we only consider the case when both the endpoint and its remote endpoint (e.g. the server it is receiving a service from) have upgraded to Multipath TCP. Spreading traffic over multiple tunnels is trivial with MPTCP: each tunnel will be treated as a different path and MPTCP will create a corresponding subflow. If certain tunnels do not work, MPTCP will simply move traffic onto tunnels that do work.

It is plain to see that, as long as there is some sort of connectivity allowed, ninja tunnels will provide reachability to any application. As the set of tunnels can be constantly evolved and personalized per endpoint, network operators will have a difficult time identifying groups of tunnels that are related and an even harder time deciding which ones to drop. We do not claim it is impossible for operators to block such tunnels, it will just be much more expensive for them to do so. Rate-limiting certain tunnels, as done with Skype today, will also not work, as MPTCP will push more traffic through the other tunnels. Finally, the Multipath TCP checksum will also detect payload changes on the different subflows, and MPTCP will terminate the corresponding subflows when changes are detected, therefore ensuring the integrity of the payload.

Efficiency. A natural concern regarding ninja tunneling is efficiency: many of these tunnels (e.g. covert channels) have a high framing overhead and have a poor ratio of useful traffic to total traffic. As the bottleneck in most networks is the user's access link (e.g. cellular connection or DSL line), using a mix of inefficient tunnels will simply decrease the total goodput of the user. Ideally, we would like the endpoint to utilize the most efficient tunnel at all times. Determining this in advance is not possible because network operators could throttle certain traffic after a period of time (e.g. after DPI makes a decision to classify the traffic).

The Multipath TCP congestion controller actively moves traffic away from congested paths onto uncongested paths, as determined by the congestion window of each subflow [25]. Is this mechanism enough to push traffic through the most efficient tunnels? We ran experiments downloading a large file (with wget over MPTCP) over a mixture of the following tunnels, all sharing the same 10Mbps access link connecting our client to the server: UDP, TCP, HTTP, DNS. We found that MPTCP behaved correctly when UDP and DNS tunnels were used, pushing most of the traffic over the more efficient UDP tunnel and achieving a throughput of around 9.5Mbps; in contrast, the DNS tunnel only achieves 7Mbps.

However, when using TCP-based tunnels (e.g. TCP or HTTP) in conjunction with DNS, MPTCP pushed a lot of traffic over the less efficient TCP tunnel, as shown in the figure 4.1.a. The problem is that the TCP tunnel hides the losses to the MPTCP congestion controller and increases delay in the process. The effect is that the MPTCP congestion controller does not move traffic away from the TCP tunnel. To avoid this problem, we implemented a technique that sets an ECN mark in the MPTCP subflow whenever the tunnel experiences a

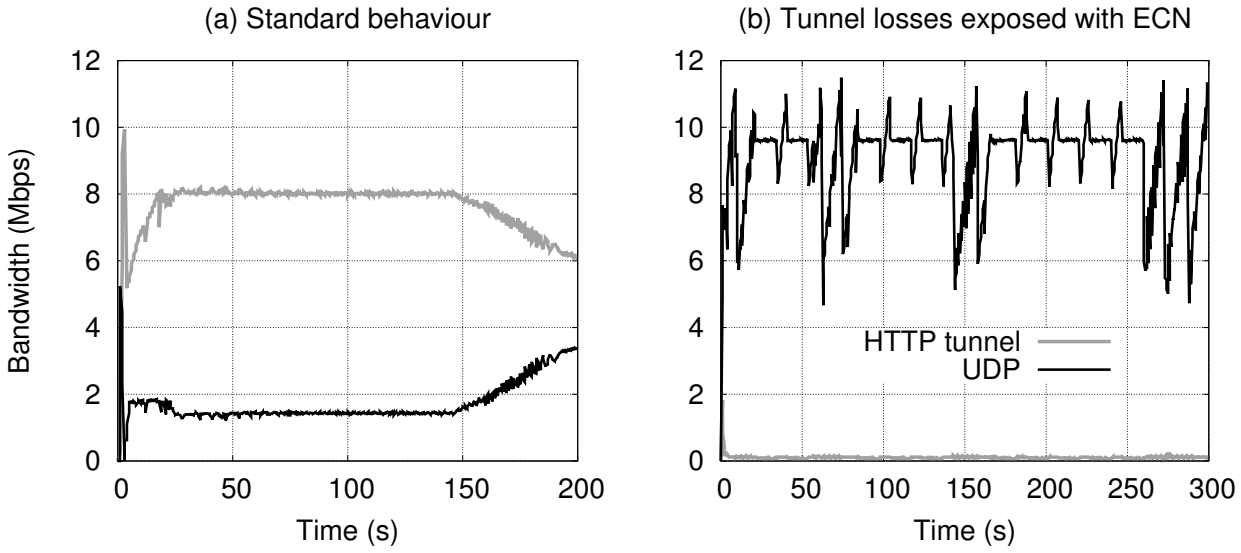

Figure 4.1: Runnning MPTCP over a UDP and an HTTP tunnel sharing a 10Mbps link

loss. This technique achieves its purpose, allowing MPTCP to balance traffic to the most efficient tunnel, as shown in Fig. 4.1.b.

4.3 A constructive approach: an explicit interface to allow endpoints and the network to communicate

In this section, we present an alternative constructive approach that assumes the cooperation of endpoints and operators. The work described below has been performed as part of the CHANGE FP7 project (2010-2013) by Trilogy 2 partners UPB and NEC, and we present it briefly here to give an overview of the whole solution space, and to show the synergies with solutions such as ninja tunneling that are born in Trilogy 2.

Assume the network operator and the endpoint (apps) want to collaborate. What is an appropriate API that would solve the problems we've outlined so far? The API must offer primitives that allow endpoints query about reachability and packet modifications, while at the same time allowing the operator to maintain its internal network topology and configuration privacy. To alleviate privacy concerns, our approach aims to allow queries whose answer the clients can find out anyway by using active probing. The usefulness of the API is that it gives definite answers quickly, rather than having to wait for an arbitrarily long probing process to finish. Our API allows endpoints to ask their network operator questions using the following syntax:

```
reach <node> [flow] -> <node> [flow] [const fields]
```

In the syntax above, `node` describes the source or destination of traffic and can be: an IP address or a subnet, the keyword *client* to denote subnets of the operator's residential clients, or the keyword *internet* to refer to arbitrary traffic originating from outside the operator's network.

The `flow` specification uses *tcpdump* format and constrains the flow that departs from the corresponding node. By altering the `flow` definition between the source and destination nodes we can specify how the flow

should be changed between those nodes.

Using this syntax clients can express how they would like the network to behave without actually knowing the network topology or the operator's own policy. For instance, the client below expects that Internet UDP traffic can reach its private IP address on port 1500:

```
reach internet udp -> client dst port 1500
```

The `const` construct allows users to specify invariants: packet header fields that remain constant on a hop between two nodes. For this, the user adds `const` and the header fields, in tcpdump format, that should be invariant. In the example below, the client specifies that the payload should not be modified in the operator's network:

```
reach internet tcp->client src port 80 const payload
```

The operator's reply to these client questions is binary, indicating whether the property holds for the *client's traffic as it passes through the operator's network*. Note that there is no guarantee the traffic will not be modified outside the network operator's domain; however, most transparent middleboxes today are deployed at the "first-hop" operator while the backbone is mostly middlebox free: by asking their first hop operator the clients can be reasonably sure their requirements hold on the whole end-to-end path.

4.3.1 Implementing the API

We can implement the API by deploying a controller in the operator's network that accepts requests from authenticated clients. The controller knows the topology of the operator, including a snapshot of router forwarding tables and the deployed middleboxes. In our implementation, the middleboxes are implemented as Click modular router [16] configurations. A configuration is a directed graph of Click elements which are processing units performing a simple task such as decreasing TTL or filtering certain packets. We have manually modeled the behaviour of individual Click elements. To answer client requests we use Symnet, a static analysis tool that applies symbolic execution to networks [23].

The controller runs client reachability checks as follows. It first creates a symbolic packet using the initial flow definition or an unconstrained packet, if no definition is given, and injects it at the initial node provided. The controller uses SymNet to track the flow through the network. The output of SymNet is the flow reachable at every node in the network, together with a history of modifications and constraints applied to each packet field. The controller then checks reachability constraints by verifying that the flow spec provided in a given node matches the one resulting from symbolic execution. The requirement is satisfied if there exists at least one flow (symbolic) that conforms to the verified constraints. To check invariants, the controller simply checks whether the field value was not modified on any possible path between the source and the destination nodes.

SymNet helps find a yes/no answer for every client question, but the operator can go further: if the answer is negative, it can reconfigure its middleboxes dynamically to honour the client's request, and then give a

positive answer. The API effectively tells the operator *what* the client wants, an information that is not available today.

4.3.2 Use cases

Client apps can use the API to quickly decide what protocol to use, including port numbers, as well as deciding whether checksumming is needed. Below we discuss two use cases we have implemented that we believe showcase the usefulness of the API.

Protocol Tunneling Say we wish to run SCTP (or any other new protocol) over the Internet. Deploying it natively is impossible because middleboxes block all traffic that is not TCP or UDP. Thus SCTP must be tunneled, but which tunnel should we use? UDP is the best choice, but it may not work because of firewalls that drop non-DNS UDP packets. In such cases, TCP should be used, but we expect poorer performance because of bad interactions between SCTP's congestion control loop and TCP's.

SCTP has to be adaptive about the tunnel it uses: first try UDP and fall back to TCP if UDP does not work, but to make the decision we need at least one timeout to elapse at the sender—three seconds according to the spec. Instead, the sender can use the API to send a UDP reachability requirement to the operator network. This request takes around 200ms to answer in our implementation, allowing the client to make the optimal tunnel choice much faster.

HTTP vs. HTTPS Mobile apps often tunnel their data over HTTP to communicate with their servers because it just works, but application optimizers may alter HTTP headers (e.g. accepted-encoding) or the payload itself (compression), breaking the application's own protocol. Should the applications use HTTPS instead to bypass such optimizers? We have measured the energy consumption of a Galaxy Nexus phone while downloading a file over WiFi at 8Mbps. The download times are almost identical, while the energy consumption over HTTP was 570mW and 650mW over HTTPS, 15% higher. The added cost of HTTPS comes from the CPU cycles needed to decrypt the traffic.

Smaller energy consumption is a strong incentive for mobiles to use HTTP, but this may break apps, so we are stuck with the suboptimal solution of using HTTPS. Instead, the client should send an invariant request to the operator asking that its TCP payload not be modified.

4.4 Ubiquitous encryption to regain control of data

Another way that endpoints can regain control over their traffic and protect their packet against middleboxes is to simply encrypt all traffic.

The motivation is to achieve by default encryption and integrity protection of Internet traffic, so that most of the traffic that flows over the Internet benefits from these capabilities.

As a large fraction of Internet traffic today is TCP traffic, the scope of this work is to produce the required mechanisms and protocol extensions to provide unauthenticated encryption and integrity protection of TCP streams, including the mechanisms for unauthenticated key exchange. While we acknowledge that there is

a significant amount of traffic that uses UDP and other transport protocols, these are out of the scope of this discussion.

Unauthenticated encryption and integrity protection provides less security than authenticated encryption and integrity protection but still provides better protection than clear text communication. The motivation for aiming for unauthenticated encryption and integrity protection is that it has a much more powerful deployment model, as it is possible to make it transparent to upper layers and users.

Unauthenticated encryption and integrity protection is also a building block for more secure communication schemes. In order to enable those, the solution for unauthenticated encryption and integrity protection must provide the hooks for external authentication. However, authentication mechanisms are out of the scope for this discussion. TCPINC is a generic approach to provide unauthenticated encryption and integrity protection to TCP streams. tcpcrypt is a specific instance of TCPINC, but in this section, where we analyze the architectural implication and broad design choices, we will analyze the general concept, hence we refer to it as TCPINC. We analyze how a TCPINC approach would interact with different components of the Internet, in other words, how would such a solution fit in the overall Internet architecture.

4.4.1 TCPINC and Upper layers

The goal is that TCPINC will be usable by unmodified applications. From the perspective of the application, it must look like they are using a regular TCP socket while in fact they are using TCPINC. This creates a powerful deployment model as applications need not be modified to use the new protocol, making TCPINC naturally compatible with existing applications.

There is however the possibility of creating an extended API that would allow TCPINC aware applications to have a richer interaction with TCPINC. There are several notable aspects to this interaction that deserve more discussion:

First, to allow an application to express that it doesn't want to use TCPINC and that it woild rather use regular TCP instead for this connection. Second, to allow an application to provide the means for authentication. Third, to allow an application to avoid reusing crypto material to avoid fingerprinting from the server. These aspects are discussed in detail in later sections.

Once all the aforementioned open issues are cleared, we will be able to properly define the capabilities of the extended API.

4.4.2 TCPINC and other security protocols

As it is a generic approach for unauthenticated encryption, TCPINC will need to co-exist with other security protocols.

4.4.2.1 TCPINC and TLS/SSL

TCPINC and TLS complement each other. TLS provides higher security as it provides authentication, encryption and integrity protection. Applications that want security usually use TLS (or other security protocols they choose). The main challenge is that its adoption requires changes in the applications i.e. they need to

explicitly use TLS. In addition, it requires some form of configuration of the authentication means e.g. certificates. TCPINC on the other hand, provides less security than TLS, but it is completely transparent to the applications. The application continues to use a regular TCP socket for its communication and it is unaware that TCPINC is running underneath instead of regular TCP. It is then targeted to those applications that for whatever reason have not been updated to use TLS or other security protocol. This provides a powerful deployment model that significantly contributes to the goal of having pervasive encryption. As TCPINC does not provide authentication, it does not require any form of credential/password/certificate configuration, simplifying its adoption.

4.4.2.2 TCPINC and TCP-AO

TCP-AO is a TCP extension to provide authentication and integrity protection to TCP segments. TCP-AO does not provide encryption or a key exchange protocol and hence it relies in other protocols to do that. The key differences then are that TCPINC provides encryption and integrity protection but does not provides authentication and that in order to be fully automatic TCPINC is fully integrated with a key exchange mechanism. This key exchange mechanism supports unauthenticated key exchange.

That being said, it may be possible for TCPINC to use TCP-AO as a building block to provide the integrity protection. In other words, further analysis is required to understand if it is possible to use TCP-AO with unauthenticated keys to provide the integrity protection and add the encryption capability and the unauthenticated key exchange.

However TCP-AO has made a set of design choices that appear to make it incompatible with the goals of TCPINC. In particular, TCP-AO cannot interoperate natively across NATs as it protects the IP addresses and the TCP ports. This seems at odds with the pervasive encryption goal of TCPINC.

4.4.3 Compatibility with TCP

Backward compatibility of TCPINC with TCP is achieved by falling back to regular TCP when one of the endpoints does not support (or is unwilling to use) TCPINC. Basically, the desire to use TCPINC in a given connection is signaled by including a new option in the SYN message. If the option is not supported, then it won't be included in the SYN/ACK, and the initiator will fall back to regular TCP and continue the communication.

The main concern with this approach is that it is vulnerable to downgrading attacks. That is, if an on path attacker removes the aforementioned option, the initiator will fall back to plain TCP even though both endpoints support TCPINC. It is not obvious how to deal with this attack other than policy i.e. to configure the node to only support TCPINC connections. This would be impractical during early deployment, but it may be reasonable once the protocol is widely adopted. One intermediate step would be to allow TCPINC hosts to remember which other endpoints support TCPINC and for those only accept TCPINC connections. This of course may be challenging in the presence of NATs (especially with Carrier Grade NATs).

4.4.3.1 Simultaneous open

An additional issue to consider is the case of simultaneous open, and whether this can be supported by TCPINC. If not, it is possible to fall back to TCP in this case. It is likely that this also depends whether crypto material is cached in both endpoints from a previous TCPINC connection or not. Further discussion and thoughts are needed in this space.

4.4.4 TCPINC and middleboxes

TCPINC is part of the ongoing efforts to harden the Internet and in particular to obtain a widespread adoption of encryption. One of the main drivers is to limit the impact of middleboxes on network traffic.

4.4.4.1 TCPINC and NATs

TCPINC aims to allow encryption of TCP streams by default. Considering the widespread of NATs in multiple flavours (NATs, NAPTs, PATs, CGNATs and probably others), it is mandatory that TCPINC is compatible with NATs (at least with a large part of the installed base of NATs). Since there is a large variety of NAT implementations out there, it is a hard task to define exactly what NAT compatibility means. However, the BEHAVE WG has defined a set of requirements for NATs (RFC4787, RFC5382 and RFC5508) and therefore at least TCPINC must be compatible with those.

NAT compatibility can be achieved natively (i.e. TCPINC wouldn't protect the affected fields) or using some NAT traversal add-on (like a UDP encapsulation). Having a native NAT compatibility is more efficient (i.e. no double headers) but results in less protection as the IP address and TCP ports are not protected. In addition, in the case that neither the IP addresses nor the TCP ports are protected, then less information is available at both ends to generate keys, which may have a negative impact in the quality of the keys if an alternative source of information for key generation is not available. This is basically a design choice between optimizing for the more common case or supporting additional security for a less common case. This is an open discussion at this point.

In any case, it seems clear that to provide native NAT traversal support, TCPINC cannot protect the destination IP address fields of the IP header nor the destination Port field of the TCP header. If both endpoints of the communication are behind their respective NATs, then it is not possible to protect the source and destination IP addresses or the source and the destination TCP ports.

A special case that seems worth mentioning is the case of NAT64. NAT64 translates the IPv6 header into an IPv4 header and vice-versa. If compatibility with NAT64 is desired, then neither the source nor the destination IP address can be protected. The destination port could still be protected. Translation of other fields in the TCP and IP header may be compatible with protection except for a few corner cases where the translation doesn't copy the value as is.

4.4.4.2 TCPINC and other middleboxes

TCPINC and TCP proxies: It seems that TCPINC could be compatible with TCP proxies that terminate the TCP connection, since TCPINC uses unauthenticated encryption. This would allow the TCP proxy to

terminate the TCPINC connection by using its own keys. As a side note, the unauthenticated nature of the solution makes it simple to provide proxies that perform TCPINC on behalf of legacy hosts.

TCPINC and middle boxes that perform segment coalescing: There are two considerations to take into account. First, if the TCP header is protected, since these middle boxes alter the header by merging two segments, the resulting header will not pass the integrity check. Second, the middlebox will merge the two payloads of the two segments. There is no well defined way to coalesce TCP options though. It depends then whether the MAC is carried in an option or in the payload. If it is carried in the payload, then it is likely that it is possible to carry enough information so that each part of the new segments can be verified with its own MAC information. If it is carried in the header, it will depend on how the middle box merge the options of the initial segments and whether there is enough space in the final segment option to include both initial options. It is less likely TCPINC will be compatible with these middle boxes if the TCP header is protected and if the MAC is carried in TCP options.

4.4.5 TCP header protection

TCPINC must provide encryption and integrity protection to the TCP segment's payload as it is the main feature it provides. It is open for discussion whether it should also provide integrity protection to (some of) the fields of the TCP (pseudo) header. First, we note that as discussed in Section 4.4.4.1, integrity protection of IP address and TCP ports would prevent native compatibility with NATs and would impose some form of NAT traversal technique such as UDP encapsulation, which would result in additional overhead. It would be possible to protect the other fields of the TCP header and this would indeed provide enhanced security for TCP. [10] performs a systematic TCP security assessment and identifies different attacks resulting from faking the different TCP fields. Note that the aforementioned analysis also includes attacks against the TCP options, so it should also be considered as part of this discussion if it would be worthwhile protecting the options as well. In any case, there seems to be potential gain in securing the TCP header fields. The concerns expressed so far about securing the TCP header are: first, how this would interact with some middleboxes (like boxes that coalesce segments) and second that this is related to where to include the MAC information in the TCP segment. If the MAC is included in the payload, then pure ACKs must include payload information, which would consume SEQ number space and we would need to eventually retransmit pure ACKs, which seems to be heading in the wrong path. If the header is not protected, this particular problem about including the the MAC in the payload goes away. On the other hand, if the MAC information is included in a TCP option, then this particular argument against securing the TCP header goes away.

4.4.6 Use of option space

TCP option space is a scarce resource as there are several widely used TCP options. TCPINC will likely need to use TCP option both in the SYN and SYN/ACK for initial negotiation and also in the data segments.

Options in the SYN and SYN/ACK. The definition of options to be carried in the SYN and SYN/ACK are the most trouble some for two reasons, namely, first because there are already many of them defined and that are

widely used, so there is not too much space left and second, because it is challenging to define a mechanism to extend the option space in the SYN and SYN/ACK that is backward compatible. TCPINC will need to carry at least one option in the SYN and SYN/ACK to indicate that TCPINC is supported and willingness to use it. If in-band key exchange is used, then the initial packets need also to carry the crypto negotiation and material. However, the crypto material is likely to be too big to carry it in the TCP options, so it is likely that an in-band approach will need to carry the crypto material in the TCP payload and second, the material not necessarily need to be carried in the SYN and SYN/ACK messages, it can be carried in the third and fourth messages for example. So, overall, it seems that TCPINC will need to carry at least an option indicating the support to TCPINC and it seems that this would be possible given the current usage of TCP options.
Options in the rest of the segments. TCPINC will need to carry a MAC for every segment to provide integrity protection. This MAC can be carried in the payload or as an option. Carrying it as an option would require every packet to carry the MAC option. The length of the option is conditioned by its crypto properties, but TCP AUTH uses a 16 byte option and this is compatible with current option usage (see [24] for more details). Besides, it seems more feasible to extend the option space for subsequent segments than for SYN and SYN/ACK so if needed this can be done. It is then an issue open for discussion if the MAC is carried in the payload or in a TCP option as both seem feasible.

4.4.7 Disabling encryption

As we mentioned earlier, TCPINC will be susceptible to be used by unmodified applications, by presenting a regular TCP socket interface. This begs the question whether it should be possible for an application to use regular TCP i.e. to avoid encryption and integrity protection. There are two options to allow an application to avoid encryption: to allow an application to select for each connection whether to use TCPINC or regular TCP or to allow to disable encryption during the lifetime of a TCPINC connection. Let's consider each of these possibilities.

Selecting TCPINC or TCP on a per connection basis. This basically means that it is possible for an application through an extended API to select whether to use TCP or TCPINC. Another way to achieve a similar result would be that implementations implement some form of policy similar to IPSec where is is possible to tell that a given set of connections should use TCPINC while another set of connections should use TCP. The motivation for doing this include:

Initial deployment and trials It is unlikely that the deployment of TCPINC will require that once TCPINC is installed in a host then all connections will be using TCPINC, especially during the early days when the implementation and specification are still evolving. It is reasonable to expect that during the initial phase at least, it will be possible to configure when to use TCPINC and when to use TCP.

Avoiding redundant encryption and integrity protection For example, some people may consider that running TCPINC below TLS is redundant and should be avoided for performance reasons. If that is the case,

then it should be possible for TLS to express that TLS connections will use TCP. Similarly, some people may consider that in certain environments (like the local network) encryption is not worthwhile and would like to use TCP for their local communications.

Disabling encryption during the lifetime of a TCPINC connection. Another possibility is to allow TCPINC to disable encryption during an ongoing TCPINC connection. It is much less clear what would be the motivation for this option. It would provide more flexibility (e.g. it would allow to use TCPINC protection while doing the TLS handshake and then disable TCPINC encryption and use TLS encryption once enabled). One possible way to implement this would be to renegotiate the keying parameters during the connection lifetime (which is probably needed anyway to allow rekeying for long lived connections) and use the NULL parameters to effectively disable encryption. The potential issue with this approach is that it may be used by an attacker to launch a downgrading attack (i.e. an attacker may be able to use this to disable encryption). Building in the features needed for protection from downgrading attacks may include additional complexity to the rekeying mechanism and to the the TCPINC protocol overall. A related issue is the possibility to upgrade the connection i.e. to start a TCPINC connection with encryption disabled and them enable encryption during the lifetime of the connection. However we are uncertain what would be a use case for this.

4.4.8 Crypto Agility

Support for agility in the cryptographic algorithms is recommended in order to be able to change the cryptographic algorithms in the case that a vulnerability is discovered for the ones currently used. In particular RFC4270 recommends: “Any new protocol must have the ability to change all of its cryptographic algorithms, not just its hash algorithm.” Related to this is the number of cryptographic algorithms that need to be initially supported. One option is that from the beginning more than one algorithm is supported, one working as the default algorithm and the other(s) working as a pre-deployed backup, so that in case a vulnerability is detected in the currently used one, it is possible to switch to the another one without deploying new code.
The counter argument is that cryptographic agility introduces complexity and complexity usually increases the possibilities of errors in the implementations, resulting in exploitable vulnerabilities. Moreover, the argument is that over the last few years, vulnerabilities related to faulty implementations have appeared much more frequently than vulnerabilities in cryptographic algorithms.

4.4.9 TCPINC and MPTCP

MPTCP as currently defined secures its signaling by including an HMAC in the relevant messages. The key used in the HMAC is exchanged in clear text during the initial MPTCP connection exchange. This implies that eavesdroppers present in the path of the initial handshake are able to see the key and then generate messages with valid HMAC that would appear legitimate to the endpoints, allowing, among other things, for an attacker to hijack ongoing communications.

The residual threats of the MPTCP protocol and potential countermeasures are described in [4]. The referenced document recommends that in order to provide additional protection to MPTCP, the payload should be protected rather than trying to secure the MPTCP signaling. This is because of NAT compatibility (i.e. securing the signaling would require securing the additional IP address and port included in the MPTCP connection and such an approach would break current MPTCP NAT compatibility). Using MPTCP in conjunction with TCPINC would allow to provide additional security to MPTCP. In particular a passive eavesdropper wouldn't be able to hijack a MPTCP/TCPINC connection. In order to succeed the attacker should be active and remain in the path during the whole duration of the attack (as he will need to change the key of one of the endpoints by its own key and needs to do the conversion from one key to the other in order to relay the messages).

4.4.10 TCPINC and TFO

After a client has established a TCP connection with a given server using the regular 3-way handshake, TCP Fast Open (TFO) allows a client to send data in the SYN for subsequent connections by presenting a cookie generated by the server and conveyed to the client in the previous connection.

In case TCPINC supports the reuse of cryptographic material, it would the naturally support TFO, since whatever material is presented to identify the cached cryptographic material can be also used as proof of a prior TCP 3-way handshake, which would then enable TFO i.e. sending data in the SYN of the subsequent connection. It seems to follow that if cryptographic material can be reused then TFO would be supported without difficulty. It should be noted that the fingerprinting concerns that emerge when reusing the crypto material are also present when using TFO, so wouldn't be a valid reason to support one and not the other one.

4.4.11 Key exchange

The goal of TCPINC is to contribute to universal encryption and integrity protection. It does so by not relying on authentication to provide either of these two features. This is because authentication usually requires more configuration from the user (i.e. getting certificates or other forms of identity proof). The expectation is that TCPINC will work automatically without requiring user intervention by default. In order to achieve this goal TCPINC must provide a form of automatic key exchange i.e. the keys used for encryption and integrity protection are exchanged without requiring any involvement of the user.

That being said, there seems to be two possible approaches to key exchange, namely in-band and out-of-band. In-band key exchange is when the key exchange is done as part of the TCP/TCPINC connection. For example, the key exchange is done in the SYN, SYN/ACK and maybe later TCP segments. Out-of-band key exchange is when the key exchange is done by a different protocol and then the generated key is passed to TCPINC for its use in a connection (similar approach as the one used in IKE and IPSec). Some arguments for the different options include:

With an in-band approach, it is not possible to secure the initial SYN and probably not the SYN/ACK either. This is certainly true as TCPINC is unauthenticated but while it is true that with an out- of-band approach it would be possible to secure the SYN and SYN/ ACK (because the key has already been exchanged), it seems

clear that it would not be possible to secure the initial messages of whatever protocol is used for key exchange (because it is unauthenticated). An analysis is needed to understand whether the lack of protection of the SYN is worse than the lack of protection of the initial message of the key exchange protocol. This is also closely related to two other undefined aspects of TCPINC: i) whether the TCP header is protected or not (if it is not protected, it is not a problem not protecting the SYN) and ii) whether crypto material is reused across multiple TCPINC sessions (if it is reused, then the initial TCPINC connection where the key is exchanged can be seen as the "out-of-band" and future connections simply rely on the existing crypto material)

With an out-of-band approach, in order for TCPINC to work, both the TCPINC stream and the key exchange stream must flow through the network. In other words, it may be possible that there are middleboxes and/or firewalls along the path that filter the key exchange protocol making TCPINC to fail even if it is not filtered. Using an in-band approach, fate sharing is achieved between the key exchange and the data stream. Related to this, it is important to understand if middleboxes and/or firewalls will allow keying material inserted in the TCPINC payload to pass.

Another argument that has been put forward in the in-band approach is that SYN processing would be increased and this may increase the possibility of DoS attacks. However, it seems that the out-of-band protocol would suffer from the same issue and it is possible to find means to limit these types of attacks (see for instance the HIP protocol).

4.4.12 Privacy considerations

Extensions to TCP to support unauthenticated encryption may have privacy implications. The first type of privacy considerations are about fingerprinting, in the sense that a responder may be able to use the TCPINC information to identify the same initiator that connects multiple times. One notable case is the one when cryptographic material is reused across multiple connections because of performance considerations. But depending on the actual design of the TCP extensions, it may be possible to use other information for fingerprinting. The second type of privacy considerations are related to a third party observer being able to identify multiple connections as involving the same two endpoints by using information conveyed in the TCP extensions. Again, one notable case is the information used when reusing cryptographic material, but there may be other type of information that an observer can use to link multiple connections. (These considerations are of course relevant for the cases where the fingerprinting nor the linkage can be done using regular TCP IP fields, like IP address and ports, notably when one of the endpoints is behind a NAT or it is changing its attachment point to the network). Additional discussion about the specific case of cryptographic material reuse are covered in the next section.

4.4.13 Reusing cypto material

Key exchange operations are costly. They are costly in terms of latency since they are performed before starting the encrypted communication. They are costly in terms of processing and bandwidth. It is natural then to try to avoid repeating the key exchange by reusing cryptographic material available between two

endpoints throughout connections. This however, has not only impact in the performance of the protocol but it also has both security and privacy implications.

In terms of security, the implications are the following. In the case of the protocol without cryptographic material reuse, the protocol is vulnerable to Man in the Middle attacks during the cryptographic material exchange. If cryptographic material is reused, this means that the vulnerability window is reduced, as the number of key exchanges is reduced drastically between two endpoints (note that in any case the endpoint should allow for changing the key material from time to time for several reasons, including privacy considerations as discussed below or re-keying). However, reusing crypto material also has the implication that if there was a Man in the Middle attacker during the initial key exchange, then the effect of the attack will not be limited to a single connection, but it will last for as long as the cryptographic material is reused. An important additional consideration to take into account is directionality. It is not clear that the security impact of reusing cryptographic material is the same when reusing the keys for multiple connection initiated by the same endpoint than when reusing the same material for multiple connections where the initiator role is changed. For example, an attacker could launch a DoS attack by initiating connections using a spoofed address and port from a well know server (e.g. it could use the IP address of a well know web server and port 80 as initiator). It then installs cryptographic material in the victim. When the victim then tries to connect to the well know web server, it will try to reuse the crypto material of the attacker, which will fail, as the server has a different key. At minimum this attack results in consuming more resources in the victim (increased latency and bandwidth). If the policy of the victim is to reject connections with different crypto material than the one cached for a given period, then the attack prevents future communications. The implications of these type of attacks are even greater when used in conjunction with MPTCP, as it is discussed in the MPTCP section. It may be worthwhile exploring the possibility of reusing cryptographic material only for connections in the same direction.

Reusing cryptographic material also has privacy implications. In terms of fingerprinting, if an initiator tries to reuse cached crypto material, the responder will be able to tell that it is the same endpoint that is reconnecting. In terms of traceability from an external third party, reusing cached crypto material is likely to require the initiator to convey in the wire some information that will allow the responder to identify the existing crypto material. This information may be used by a third party to link multiple connections between two endpoints even if the IP address and/or port has changed. (this of course depends on the details of the protocol). As the tradeoff between performance and privacy is likely to depend on the utility function of the endpoints, it is possible that a good solution is to allow the endpoints to determine if they rather reuse cached material or not for each new connection.

4.5 A thought about the future

We have outlined a number of tools that endpoints can use to regain control over the service they receive from the network. These aim to counterbalance the power network operators exercise over customer traffic

by deploying middleboxes without customer consent. Opportunistic encryption raises the cost for networks to snoop on and modify traffic.

We have further proposed two alternative approaches to liquidise access networks and allow endpoints to make the best use of their links.

Our more pragmatic solution, **ninja tunneling** picks a contract that seems reasonable for endpoints, *ubiquitous reachability and immutable payload*, and enforces it by using MPTCP to spread the data over a mix of different tunnels. The only thing an operator can now do is make it cheaper or more expensive to achieve this contract: the app does not have to worry about the details, as ninja tunneling will dynamically find the most efficient tunnel and send most data through it. Eventually, operators that force endpoints to use inefficient tunnels will lose their customers, so this may well act as a driving force towards removing unwanted middleboxes.

We have also outlined a constructive approach (proposed in the CHANGE FP7 project by the same partners) where the endpoints can use an API to query the network about the service it offers. This offers more information to both operators and endpoints, and in the long run we believe this approach is preferable for operators: they still get to run their app optimizers for traffic that allows it explicitly, but they will have to honour the requests of users asking middleboxes to not mess with their traffic.

The API and ninja tunneling complement each other: ninja tunneling is a short term fix and a stick to beat operators into being nice. The API is the proper way of implementing cooperation, but it is doomed without something like ninja tunnels that forces operators to adopt it.

5 NFV as a particularisation of the Trilogy 2 infrastructure

5.1 Introduction

Deliverable D2.1 set out the basic Trilogy 2 architecture, which we developed and contributed to the work of the ETSI NFV ISG. During this year, we have gone beyond the ETSI in terms of standardisation and have started addressing other Standards Defining Organisations (SDOs) like the IETF. In addition, we have continued a more fundamental work on the representation of resources, which is finally impacting our work on the information models for the Trilogy 2 architecture.

5.2 A generalised representation for resources in the liquid network

Within Deliverable D2.1 we set out a more formal framework which brings together the approaches of systems engineering with that of configuration and programming. This approach has played an important role in the development of NFV as it provides a formal framework for defining the relationship between VNFs and the NFVI which is hosting the VNF.

The significant point is that this relationship is not a normal functional interface of systems engineering and cannot be described as such. Indeed, in functional terms, the VNF *is* the infrastructure: the VNF is only a configuration of the hosting infrastructure function and has no separate existence apart from the hosting infrastructure.

In this section we do not repeat what is already set out in Deliverable D2.1. Figure 5.1 below reproduces the basic many to many hosting relationship between virtual functional blocks and host functional blocks from the Deliverable D2.1 framework.

This framework immediately addresses a number of axiomatic principles of the Trilogy 2 architecture including

- Portable Applications Create Liquidity
- Transport, Processing and Storage are all Strongly Inter-Dependent
- The Starting Default is Real-Time Parallel Concurrent System Specification
- Everything has Its Own Defined Lifecycle
- The Starting Default is a Decompositional Model

The axiomatic principle which is not immediately addressed by this framework is “We Must Have Measures for the Capacity of All Types of Resources”.

However, the functional block framework does provide insight into how Trilogy 2 resources can be specified by comparison and extrapolation from transport.

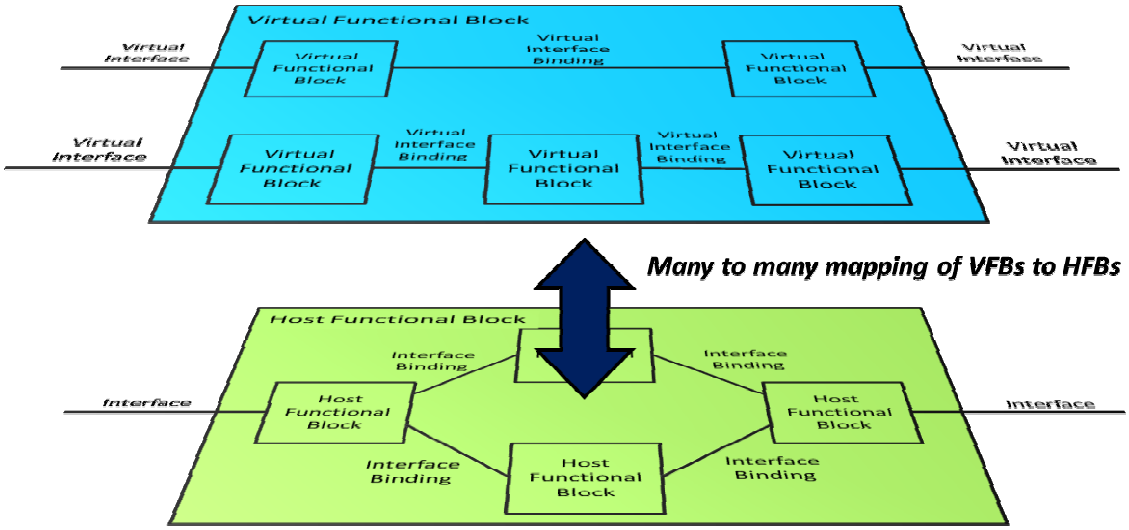

Figure 5.1: Many to many hosting relationship of the D2.1 virtualisation framework

This starts by equating a transport channel to a functional block. The transport channel has input, outputs, a static transfer function and holds dynamic state.

Trilogy 1 made implicit use of this framework and of the bit as the measure of resource capacity and application load. In order to extend this to the more general needs of Trilogy 2 , we present two stages of generalisation.

In order to explain the generalisations, the next section presents a summary of the the way we measure transport resource highlighting the features which are important to the generalisation, in particular revisiting some of the fundamental derivation of the bit as a unit of transport resource by Shannon. Following this, the first generalisation is then then presented which is to extend from a transport channel to a network. Finally, the second generalisation is discussed which is to extend from a network to a general host functional block of a hosting infrastructure of which the NFVI is an important example.

While the objective is measures of generic Trilogy 2 host functional blocks, the intermediate analysis of the network plays three useful roles. First, there is strong linkage with Trilogy 1. One of the key successes of Trilogy 1 was the generalisation of transport control to a network wide context and to optimise the total network resources across all the end to end transport sessions using the network. Second, the network context requires the introduction of processing required for routing and resource allocation. Indeed, one of the essential roles of the transport layer is to completely abstract these processing functions of the network from the user and present a simple point to point channel to the user. As such, the network is a hybrid example. Thirdly, there is already developed in ITU-T a powerful framework of abstract functional blocks which describe networks. These are already the basis for most transport network management systems and are also the basis of the recently agreed Open Networking Foundation (ONF) architecture for SDN.

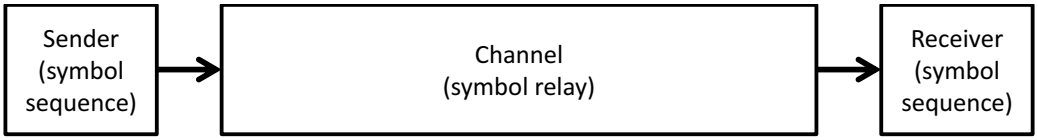

Figure 5.2: A transport channel

5.2.1 Measuring Transport Resource

In a digital world, we tend to take for granted the ease with which we measure the a) capacity of the transport channel and b) the size of the information load presented for transport: everything is measured in bits. The bit is now a ubiquitous and universal measure and essentially defines digital technology. However, the original definition of the bit developed by Shannon was not primarily intended to define any technology, and it was defined in the era of analogue telecommunications channels, telegraph text messages, and analogue voice calls. In order to present the way in which the capacity of general hosting resource can be measured, it is helpful to give a short prcis of Shannon's original work. The channel as defined by Shannon is illustrated in Figure 5.2.

There are a number of important features of the transport channel which are implied in Figure 5.2:

- the sender, channel and receiver are in continuous and open ended operation and do not "run to completion";
- the sender produces a sequence of symbols in real time;
- the channel relays symbols in real time;
- the definition of a symbol is central to the definition of a channel.

On this last point, Shannon precisely defines information as a selection from a set and each particular symbol represents a member of the set of possible selections. There is therefore no absolute meaning to any symbol and the selection of any one symbol is essentially arbitrary. Shannon essentially defines information in the context of a single operation type "select" and all information is in the form of "select symbol".

Importantly, we note that the channel only supports one basic operation - "relay symbol" (or maybe more correctly "relay select symbol") - and there is no possible choice of different operations. As we look at generalisations, we will note that general host functional blocks support many possible operations. However, for the channel, the total number of specific operations that are supported by the channel on each input event is therefore simply the number of possible symbols. With this basic model, Shannon defines separately the capacity of the channel and the information load of the sender with a view to giving an answer to the question of whether the channel capacity can support the sender's information load.

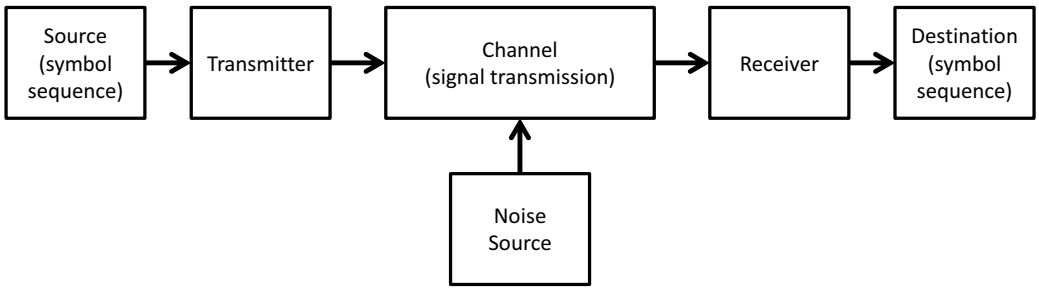

Figure 5.3: Shannon's reference architecture

- Capacity: Shannon defines the capacity of the channel as the logarithm of the total number of possible sequences of symbols that can be sent over a long period of time divided by that time. This takes account of the situation where not all sequences of symbols are possible. By choosing logarithms to base 2, the capacity is defined to be "bits" per unit time. The "bit" is simply the scaling defined by the base of the logarithms and is no more fundamental than any other base. The linkage with binary encoding is a matter of convenience, not fundamentals.

- Load: Shannon defines the information rate load of the sender as a form of entropy, based effectively on the probability of the next symbol which may depend on the history of symbols in the sequence. In order to calculate this, Shannon's model of the source is a state machine. Again by using logarithms to base 2 in the entropy calculation, the information rate is defined to be "bits" per unit time. Again, the "bit" is simply the scaling defined by the base of the logarithms and the linkage with binary encoding is a matter of convenience, not fundamentals.

Shannon's actual basic reference architecture diagram is shown in Figure 5.3. Included in this diagram are two extra functions (other that the noise source), the transmitter and the receiver. In Shannon's model, these functions are to encode the sender's symbols to the symbols of the channel and back again. Shannon does not assume that the sender and the channel have the same symbol set. The as with the sender, Shannon formally models the encoder and decoder as state machines.

A critical point which is easily forgotten in today's digital telecommunications is that Shannon does not assume the channel is a homogeneous selection of uniform symbols so that the capacity of the channel is available to any load. The available capacity is only available if the rate at which a particular channel symbol is chosen is inversely proportional to the time taken by that symbol in the channel.

Shannon's basic thesis is that *it is possible to devise a encoding* such that a sender's message can be carried if the sender's information rate is less than the channel capacity. The exploitation of the channel capacity depends on encoding but optimal encoding is possible.

- The use of binary encoding for both the sender and the channel (as is effectively the universal solution today) is a simple and practical way of ensuring that the full channel capacity is always available to every sender without any encoding mismatch.

- Shannon assumes that the encoder can construct sequences of “select symbol” for different symbol sets which are operationally equivalent. For example, the sequence “select 1” “select 1” “select 0” “select 1” using the binary symbol set he assumes can be made equivalent to “select D” from the hexadecimal symbol set. Importantly, if they are equivalent, then the encoding is reversible and so decoding is possible.

When we now identify the channel as a functional block, we see that the input of the transport channel is driven by a sender and the output is connected to a receiver. The transfer function is effectively an operation: “relay input symbol to output after transport delay”. The functional block state holds the symbols in flight between the input and output.

There are a number of points which make the representation of Shannon’s model in functional block form especially significant.

- The sender, the transmitter encoder, and receiver decoder are explicitly identified as state machines.
- By implication, the channel and receiver as also state machines.
- Every state machine defines a functional block and vice versa therefore we can be directly equate Shannon’s state machines with functional blocks

We can redraw Shannon’s diagram with this understanding and go further. We can note that the process of encoding and decoding enables the hosting of a virtual channel which is logically relaying the sender symbols. this virtual channel is hosted on the actual channel which is a host functional block. This is shown in Figure 5.4.

From this we can see that Shannon’s basic model on which he developed his measures of capacity and load have much more in common with generic processing that might be assumed from the modern perspective of digital telecommunications. There are number of observations that can be made at this point on how Shannon’s model relates to the more general model required for Trilogy 2 .

- The encoder and decoder state machines are potentially very generic and it is therefore possible to say that any functional block is essentially an encoder: essentially for any functional block, the outputs are an encoding of the inputs.
- The information on the input of a state machine/functional block can be either lost by the functional block, held in the state of the functional block, or passed to the output. If the functional block does not lose information and it has finite state, then over time the output information will equal the input information. A functional block cannot “create” information on its output.
- Time is a principle measure of cost. For a channel of a given capacity, the cost of any one usage of the channel is simply the time the channel is occupied by the usage. This directly compatible with the congestion based control mechanisms developed in Trilogy 1.

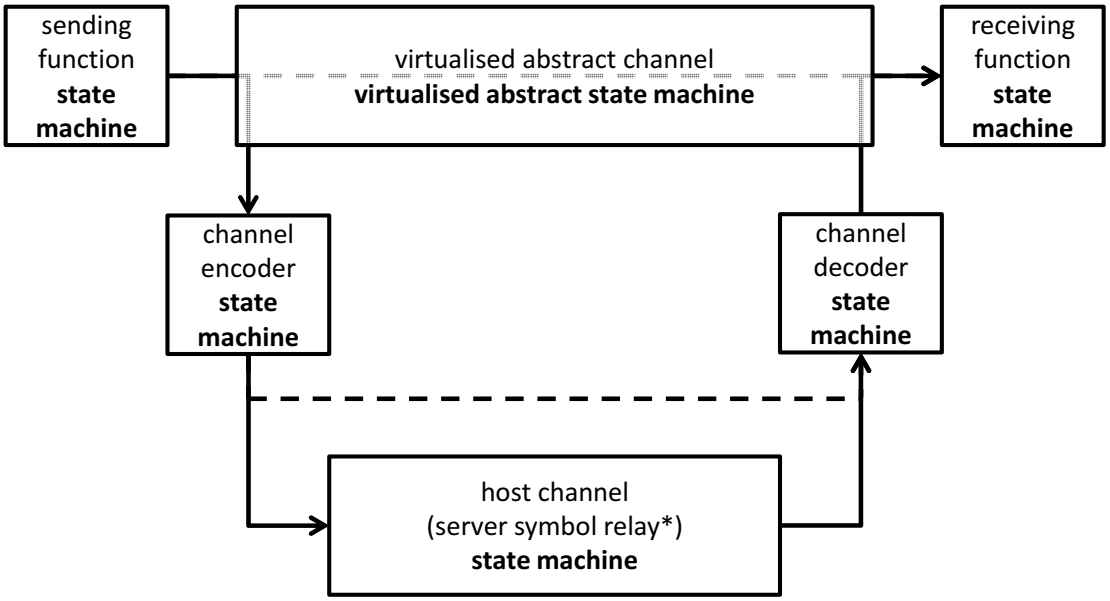

Figure 5.4: Encoding and decoding from a virtual channel to a host channel

- The measure of capacity is the number of different ways the resource can be used over a given period of time.
- Things which are predictable require little or no cost.

There are also observations which highlight the difference between the channel and the more general case.

- Encoding and decoding are a functional pair whose net effect is a null function. Ultimately, Shannon does not consider one separate from the context of the other.
- Sequences of symbols from one symbol set can be arbitrarily mapped to sequences of symbols from a different symbol and have complete functional equivalence. This is not true of general functional operations.
- A corollary of the above is that assuming compatible decoding, changing the encoding has no impact other than to change the capacity of the virtual channel. Clearly in the more general case, changing the encoding fundamentally changes the functional block.

5.2.2 Measuring Network Resource

The first extension of the basic channel as defined and modelled by Shannon is to generalise the channel to a network. This section is largely based on the transport functional architecture of the ITU-T G.800 series of Recommendations. In the case of the channel, the operation was simply “relay symbol”. In this case, the basic operation is “relay to destination symbol”. There are five immediate aspects to this extension:

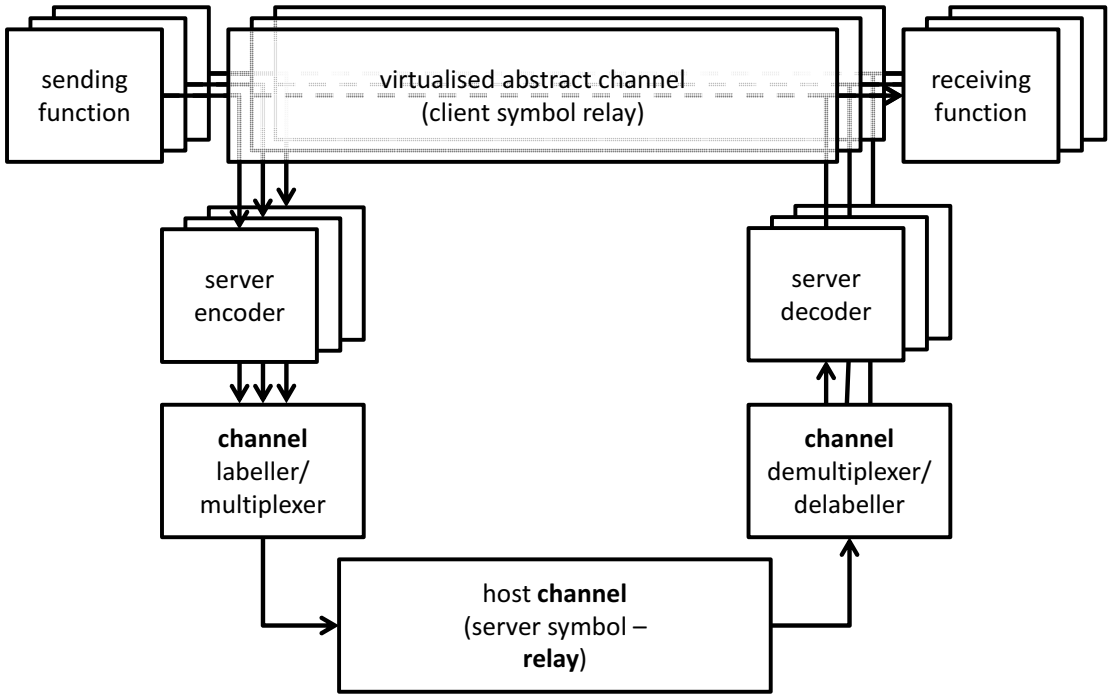

Figure 5.5: Multiplexing

- the number of possible operations for the network is now the three way product of the the number of symbols, the number of sources, and the number of destinations;
- there may be many senders' symbol sequences all using a particular host channel simultaneously and this requires a resource allocation function;
- any one sender's symbol sequence may be routed across many host channels and this requires a filtering by destination function;
- at the receiver there may be symbols from different sources all merged together and this requires a filtering by source function;
- unlike to the encoding function and decoding function of symbols, the filtering functions and the resource allocation functions stand alone and do not have a "neutralising" partner function.

An initial step in considering this extension is to consider first multiplexing many virtualised channels on the same host channel as illustrated in Figure 5.5. In order to multiplex, labelling is needed in addition on the encoding and decoding.

If encoded symbols from different senders were simply buffered in to the host channel, the receiving end would have no way of knowing which symbols belong to which sender. In order that the receiver knows to which sender's virtual channel each symbol belongs, each symbol must be labelled. This labelling is an information requirement and may be calculated as an information rate load. We can say:

- some labelling schemes, notably TDM, are fully deterministic and therefore there is no information rate and so no labelling information is required (in practice, the label is implicit in the deterministic multiplexing sequence);

- some labelling schemes are maximal entropy in that all channels are treated as equally likely for all symbols and so every possibility takes the same labelling resource;

- it is possible, and theoretically more efficient to design a labelling scheme where the the channels with higher rates of symbols use less label resource (ie shorter labels) compared to channels with lower symbol rates.

In practice the labelling is an encoding the identity of the tributary port on the multiplex function. The information is already there in that the symbols arrive through the port which automatically gives them an identity within the function. Noting this, we see that the multiplexing function has not created information. Nor indeed, has the demultiplexing function lost the information as the information after demultiplexing is present in the trib port of the demux function.

In addition to adding labelling, the multiplexer must also allocate the capacity of the host channel to the symbols of the client channel. In contrast to the allocation of the capacity in the case of an unmultiplexed channel which has a single input, this capacity allocation function has a different independent input from each virtual channel. The allocation function is likely to require to know the load arriving from other inputs in order to decide on its behaviour. This shared common state suggests that localised implementations are likely to be more efficient that physically distributed solutions where there is a significant delay in acquiring the common state. This is illustrated in Figure 5.6.

The second stage of the extension is to extend the single host channel to a full network with topology. The sender, present at one source has messages for different destinations. However, in addition, the different destinations will have messages from different sources. The symbols at the senders' ends must be labelled according to their destination while at the receivers' ends must be labelled according to source in order to reconstruct the individual messages from individual sources. In other words, the context of the labelling must be extended from a single host channel to the network with topology.

Figure 5.7 shows a generic representation of a network with the labelling and filtering that must occur in order to get one sender's messages to the appropriate destination and then disentangle this message from other messages at the far end. Note, this has been developed from simple information theory and has made no reference to any technology or implementation. This is the techniques that was followed by the ITU-T in developing the G.800 series of recommendations and ensures the functional model is truly abstract and independent of implementation. This is very closely associated with the portability requirement of Trilogy 2 .

We can now look at the network from the outside. The network is in fact a host function which is capable of supporting many virtual channels between senders and receivers at its end points as illustrated in Figure 5.8.

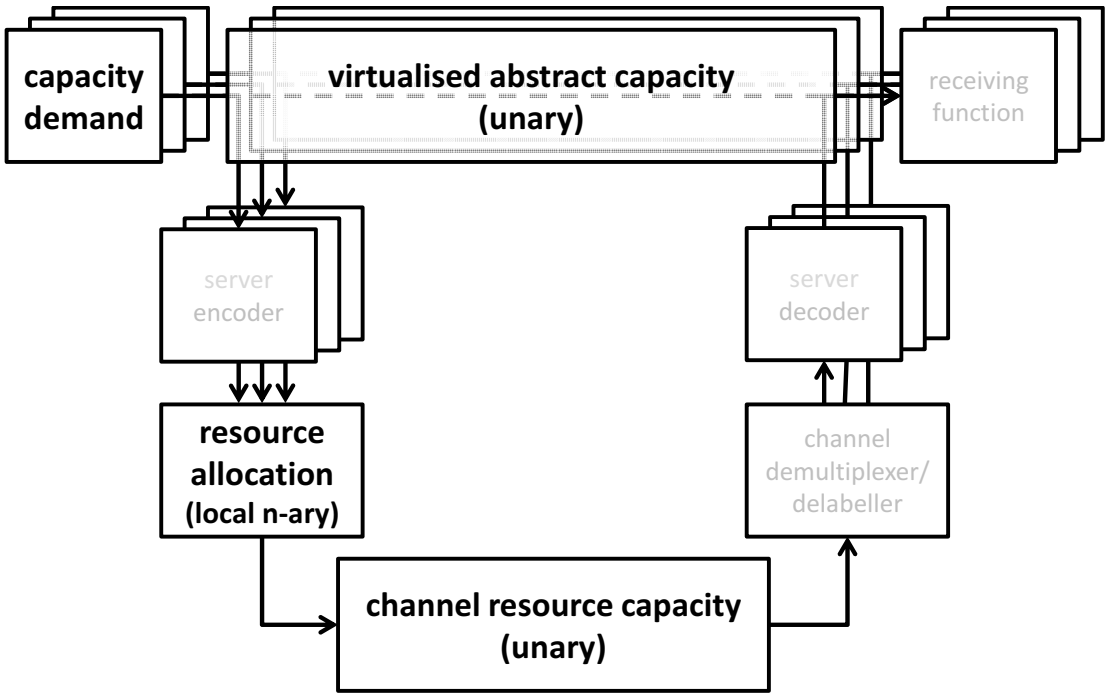

Figure 5.6: Resource allocation

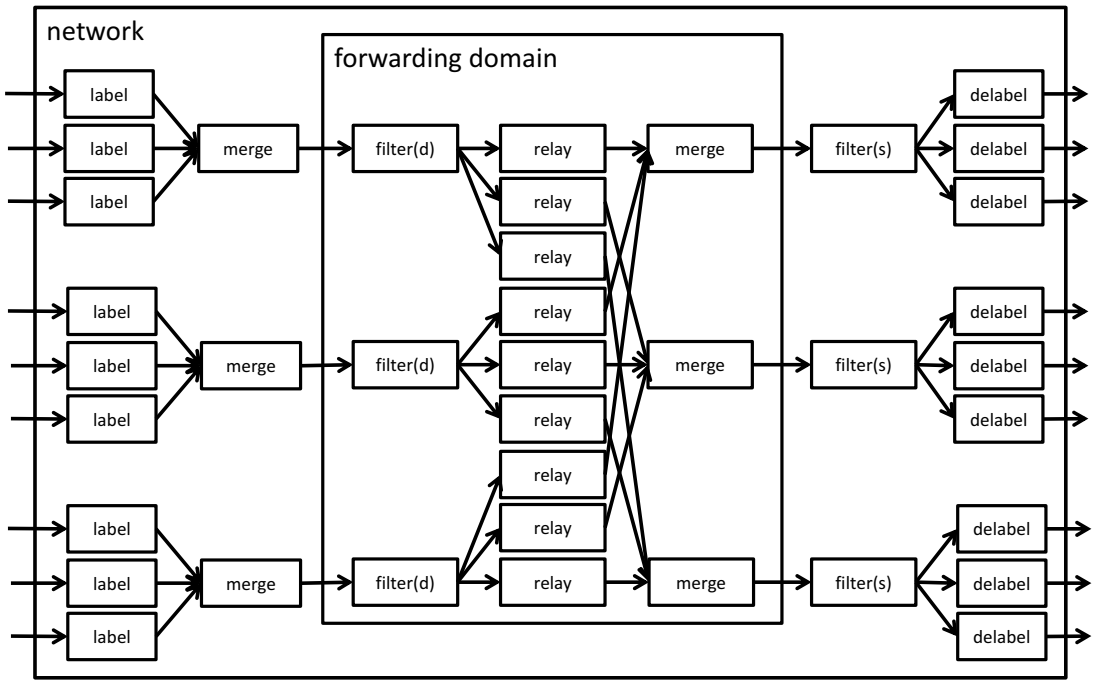

Figure 5.7: A general network

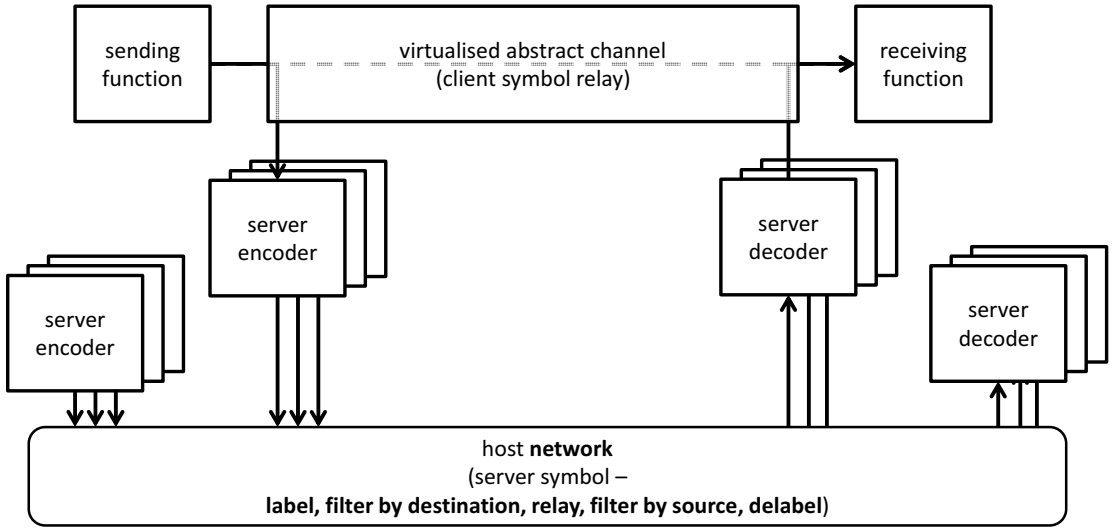

Figure 5.8: Using a general network to support virtual channels

Having extended the general functional model of a channel to that of a network, we are left with the question of "what is the capacity of the network?".

In any practical network, that will depend on the routing of the traffic. The capacity must include the "to destination" part of the the "relay to destination symbol" operation.

However, this is consistent with Shannon's original definition of capacity which is the logarithm of all possible sequences of operations which can be supported over a long period of time divided by that period of time. Normally, much longer sequences of operations are possible to some destinations than others.

This is a result of the fact that most networks are cost optimised around popular routes. However, this is exactly the same observation that Shannon made about the optimisation of codes like Morse Code around the most popular letters of the alphabet.

Therefore the Shannon capacity of the network is the maximum capacity of the network assuming destinations are chosen exactly according to the topology of the network. If the actual demand deviates from this optimum, the maximal capacity cannot be achieved.

However, we also note that unlike the simple "select symbol" operation, an encoding of a sequence of popular destinations cannot be made equivalent to a less popular destination. There is therefore no equivalent to Shannon's main theorem of the possibility of ideal encoding.

We can make one important conclusion at this stage. We note that when we consider partitioning of a network according to the rules of G.800 series, we see routing is a sequence of "relay to destination symbol"

across subnetworks which are bound into the network topology. We can say that the native language of the network for routing is sequences of intermediate subnetwork destinations. The network therefore must create an encoding of each network destination, in the context of each source, into a sequence of intermediate subnetwork destinations, its native language. As with Shannon encoding, the efficiency of the use of the network is determined by the efficiency of this encoding.

At intermediate subnetwork destinations in the network, it is possible to consider that an encoding of a sequence of popular intermediate subnetwork destinations can be made equivalent to a less popular one. This is exactly what MPTCP does in forming multiple routes. This suggests that Shannon capacity and information are an equivalent way of analysing the network resource optimisation. MPTCP is a mechanism which can allow the encoding of sequences of intermediate subnetwork destinations which define multiple paths in order attempt an optimal match between the network capacity and the traffic load from all senders.

As a final observation on routing, in the same way that binary makes possible a maximal entropy channel were the capacity is the same no matter what the input bit sequence, the same is possible topology. This is the so called "fat tree" topology which we now see in widespread use in data centres. The fat tree has the property that with the constrain that senders' and receivers' port capacity is available, all possible traffic matrices can be supported with equal capacity. In datacentres and in other places, this independence is more important than cost optimising to popular destinations.

As a final note on this definition of network capacity, the definition did not address what should be the base of the logarithms. In this case, base 2 does not seem to be the natural answer. Base 2 works with a channel on the basis that optimal encoding is possible and therefore the size of the symbol set does not matter. However, in the case of destinations, the set of destinations does matter and so choosing the base to take account of the size of the destination set seems more appropriate.

5.2.3 Initial Suggestion on Measuring Trilogy 2 or NFVI Resource

This work is still at a relatively early stage and so this section sets out the general elements of the possible direction to measuring the capacity of general host functions and the load placed on then by virtual applications (in the case of NFV, VNFs).

Figure 5.9 illustrated the general scenario of a virtual functional block being hosted by a hosting functional block. The interface to the virtual functional block is actually an interface to the host functional block which realises the operations of the virtualised functional block.

However, the host functional block has two types of interfaces. In addition to the operational interfaces, it also has configuration/programming interfaces. This is the type of interface through which the host function can be configured to realise the virtual functional block with its operations. This was more formally developed in D2.1 and Figure 5.10 is reproduced from D2.1 and shows how these two types of interface effectively freeze some state in the host function block in order to realise the virtual functional block. (This figure also identifies a third type of interface for the private management of the host functional block).

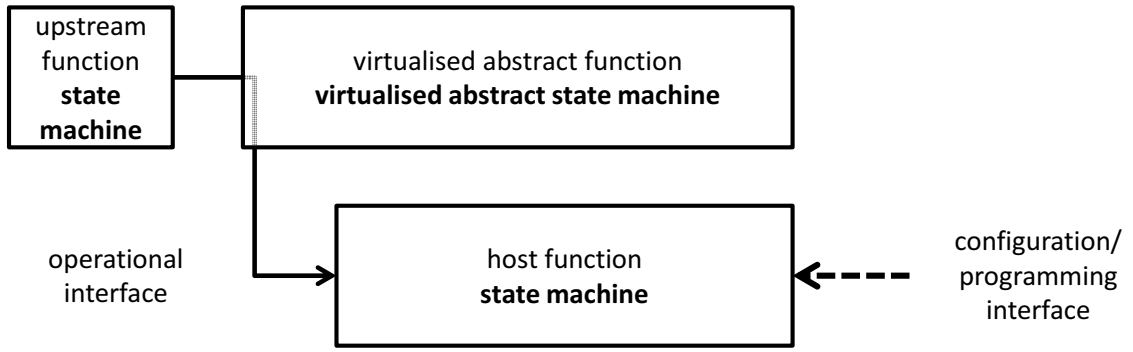

Figure 5.9: General hosting of a virtual function

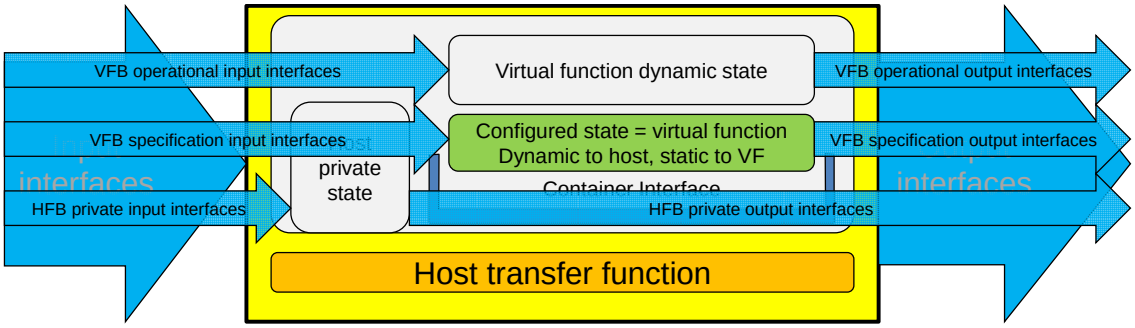

Figure 5.10: Interfaces of a host functional block

The basic question is “what is the capacity of the host functional block?”.

The suggestion is to again reuse Shannon’s definition of capacity. The capacity of the host functional block would therefore be defined as the logarithm of all the possible sequences of operations that can be supported from all possible virtual functional blocks that can be hosted over a long period of time and divided by that period of time.

However, as we noted with the network, the number of operations in any one sequence will depend on the time taken to perform the operations, and this can vary greatly. Moreover, some hosts will be tailored to support some operations more quickly than others. Even more, many hosts are only capable of supporting certain types of virtual functional blocks.

This latter point is important as hosts cannot be restricted to “universal machines”. There are many useful host devices which are designed to host a certain class of virtual function. there are two obvious examples in the case of a network hosting only virtual channels and in the case of storage arrays only hosting virtual storage. However, even “processing” hosts may not be universal. This restriction is even more important when we consider that in the real-time environment, a host is only supporting a virtual functional block if it meets all its timing requirements of its operations, not just the logic of the operations. In this regard, most “universal machines” are not at all universal.

As with the network where we noted we define the maximal capacity of the network, we also define the

maximal capacity of the general host. The efficiency of the use of the host is determined by the encoding of the virtual functional blocks into the language of the host.

As an obvious example and very practical example, consider a virtual functional block which is implemented by a programme on a standard server machine (eg x86). In this example, the maximal capacity of the host functional block is primary determined by the speed with which the Central Processing Unit (CPU) processes x86 instructions and the the encoding of the virtual functional block is the programme. The efficiency is determined by the efficiency of the programme to exploit the language of the host, that is the x86 instruction set. This accords with the current notions of both measuring the capacity of the host and the efficiency of the implementation. However, in this case, the derivation is not empirical, it is mathematical. Moreover, the derivation can be applied to all forms of host not just CPUs with an instruction set where execution rate is an obvious performance benchmark.

There is still some work needed to develop this definition as a fully operational measure of host resource. This approach appears to have the follows advantages:

- it is mathematically based;
- it fits with existing pragmatic measures;
- it directly deals with universal hosts and load optimised hosts;
- the separation of maximal capacity from the encoding efficiency gives a practical way of dealing with the wide performance differences that can exist as a result of different coding implementations.

5.3 Evolution of the standardisation landscape

The Trilogy 2 architecture described in Deliverable D2.1 is heavily based on the architecture proposed by the ETSI NFV ISG. From the moment we released that deliverable until now, we have contributed at different initiatives at the IETF and the IRTF covering gaps or enhancing out original Trilogy 2 architecture. We have had different degrees of success in bringing Trilogy 2 work into different SDOs. In some instances we are working in established IETF work groups (WGs) and in others we are in the process of establishing new ones.

5.3.1 IETF: Service Function Chaining

The SFC working group within the IETF is devising mechanisms to steer traffic in a data-centre-alike environment in such a way that elements implementing Service Functions (SFs) can be chained in a given order to result in different service function chains [11]. These SFs can be either physical devices or virtualised appliances, whereby VNFs are covered.

A key aspect when chaining SFs is how to check that the resulting function chain is working correctly. Additionally, means are needed to spot points of failure when something goes wrong. This is covered by the term “OAM” [3]. In this spirit, we are contributing to the requirement collection for an OAM framework for SFC. Initialy, our findings were contributed to [18] for the IETF#90. This Internet Draft (I-D) was merged

with [2] for the IETF#91. Once the framework is in place, we will start the work on concrete protocols by selecting existing OAM protocols and analysing the gaps to fully cover the specifics of VNFs.

5.3.1.1 Intent and scope of SFC OAM

We distinguish different types of Appliance or Service Function that can be chained in a function chain:

(i) Ingress appliances that classify packets (per tenant, per flow etc.), handle metadata generation, set up one or more chains and mark packets.

(ii) Stateful appliances such as firewalls or other devices, where packet modification and/or Layer 4-7 session termination will be performed

(iii) Transparent appliances (e.g. DPI devices, stealth firewalls, etc.) which do not perform Layer 4-7 session termination and where packet modification may or may not be performed.

Our intent is to monitor and ensure the *integrity* of the Service Function Chaining. This I-D is not attempting to monitor services or trying to replace transport OAM. It rather will capitalise on existing OAM functions in the network and define the missing pieces. a Specific challenge in SFC is that the elements of the chain are not only not required to deliver all the Internet Protocol (IP) packets they receive, but may also generate IP packets themselves.

5.3.1.2 Relationship with other Trilogy 2 activities

With the advent of Network Function Virtualisation, network functions that are traditionally implemented on specialized hardware devices are moved to virtualised computing platforms. A Virtual Network Function implements the same network function (e.g. firewall, load balancer) as its non-virtualised pendant, but is deployed as a software instance running on general purpose servers via a virtualization layer, such as an hypervisor. This way, a network service is a sequence of topologically distributed VNF instances that become a VNF chain. These VNF chains can be treated as Service Function Chains.

5.3.2 IETF: VNFPOOL

The use of VNFs opens new challenges and requirements concerning the reliability of provided (network) services. When network functions are deployed on monolithic hardware platforms, the lifecycle of individual services is strictly bound to the device availability, and management backplanes may detect outages and fail over all affected services to new instances deployed on backup hardware. On the other hand, with VNFs, individual network functions may still fail, and there are more factors of risk such as software failure at various levels, including hypervisors and virtual machines, hardware failure, and instance migration that may make a VNF unreliable. Moreover, there is currently no mechanism to provide detection and redundancy for individual functions of a VNF chain, and the use of VNFs introduces new problems in reliable service provisioning that are not addressed in existing control and management mechanisms. First of all, how to detect and respond to a failure in an element of a VNF or of a chain? And how to inform VNF chain neighbours about a failure, and how do they respond? And how to transfer VNF state information to its backup VNF?

In this context, the VNFPool effort in IETF tries to achieve higher reliability for VNFs, and in particular to adopt VNF pooling mechanisms where a number of VNF instances can be grouped as a pool to provide the same function in a reliable way. VNFPool tries to address challenges and open questions concerning the reliability of individual VNFs belonging to VNF pools: e.g. how to manage the redundancy model (e.g. select active/standby for a VNF instance in a pool) while considering the policy and the physical infrastructure conditions, and how the service states of a VNF are maintained and synchronized with backup instances in a pool.

5.3.2.1 VNFPool Requirements and Use Cases

As a complement to the VNFPool architecture exercise, some work around requirements and use cases for VNFPool in the NFV realm is active in IETF. In particular, the deployment of VNF based services requires a transition of resiliency capabilities and mechanisms from physical (specialized) network devices, typically highly available, to such entities (like VMs) running VNFs in the context of pools of virtualized resources. When moving towards a VNFPool enabled approach for VNFs deployment and operation, the generic resiliency requirements are translated into the following ones [26]:

- Service continuity: when a hardware failure or capacity limits (memory and CPU) occur on platforms hosting VMs (and therefore VNFs), it is necessary to migrate VNFs to other VMs and/or hardware platforms to guarantee service continuity without negligible impact to users

- Topological transparency: the hand-over between live and backup VNFs must be implemented in a transparent way for the user and also for the network service itself. This means that the backup instances need to replicate the necessary information (configuration, addressing, etc.) so that the network function is taken over without any topological disruption (i.e. at the VNF chain level)

- Load balancing or scaling: migration of VNF instances may also happen for load-balancing purposes (e.g. for CPU, memory overload in virtualized platforms) or scaling of network services (with VNFs moved to new hardware platforms). In both cases the working network function is moved to a new VNF instance and the service continuity must be maintained.

- Auto Scale of VNFs Instances: when a VNF requires increased resource allocation to improve overall service performance, the network function could be distributed across multiple VMs, and to guarantee the performance improvement dedicated pooling mechanisms for scaling up or down resources to each VNF in a consistent way are needed

- Multiple VNF Resilience Classes: each type of end-to-end network service (e.g. web, financial back-end, video streaming, etc.) has its own specific resiliency requirements for the VNFs that implement such service. While for operators it is not easy to achieve service resiliency SLAs without building to peak, a basic set of VNF resiliency classes can be defined in the context of VNFPool to identify some

metrics, such as: if a VNF needs status synchronization; fault detection and restoration time objective (e.g. real-time); service availability metrics; service quality metrics; service latency metrics for VNF chain components.

A set of use cases for VNFPool are under investigation in IETF [26, 15], with the aim of validating the VNF-Pool architecture concepts and also to identify any new resiliency requirement that improve the architecture specification.

5.3.2.2 VNFPool Architecture

The VNFPool architecture aims to provide reliability mechanisms to VNF instances, that are not typically considered as built-in functionalities on their hosts, i.e. the general purpose servers. Apart from the already mentioned VNF instance failures at both hardware (i.e. server overload) and software (i.e hypervisor, VM, VNF itself) levels, a further crucial aspect to be considered when providing VNF pooling mechanisms is the instance migration caused by performance downgrade by excessive load (e.g. CPU, memory, disk I/O), server consolidation or any network service constraint upgrade. Even if this VNF load balancing aspect is distinct from a hardware/software failure, it may give the same appearance from a pure VNF perspective, and must be considered in the VNFPool architecture. With VNFPool, a given group of VNF instances is called VNF set, and it can include either single or multiple types of VNF, and each type of VNF has typically a number of instances providing the same network function. The VNFPool architecture aims to provide mechanisms to dynamically manage a set of VNFs providing the same function in a transparent way for end-hosts and service control entities, and also map the current VNF in use with the group it belongs. The VNFPool architecture, as it is currently defined in IETF, is depicted in Figure 5.11. A VNF has a VNFPool associated that contains a certain number of VNF instances, called VNFPool Elements, that provide the same network function. Therefore a VNF set is transversal to the pools in Figure 5.11 and can be grouped into multiple VNFPools implementing specific VNFs. For each VNF, a Pool Manager manage the reliability of the VNF itself, by selecting the active instance and interacting with the Service Control Entity for consistent end-to-end network service provisioning. Each VNFPool is typically identified with a kind of abstract and unique identifier across all the Pools, and the role of the Pool Manager is to keep the active instance transparent to the Service Control Entity and map this identifier with that active instance.

On the other hand, the Service Control Entity is responsible for the provisioning of the network services, and therefore is the high level entity that combines and orchestrates all the VNFs under its ownership. The major benefit when using a VNFPool approach is that the reliability mechanisms and the pooling of VNFs is completely transparent to the Service Control Entity: the management of the redundancy of each VNF composing a network service (i.e. a VNF chain) is carried out inside each VNFPool by each Pool Manager. This way, a VNFPool enabled VNF is exposed to the Service Control Entity as a normal VNF. When composing network services, VNFPool enabled VNFs become part of VNF chains managed and orchestrated by the Service Control Entity, following the principles defined by IETF for the Service Function Chaining (SFC) [11]. A service

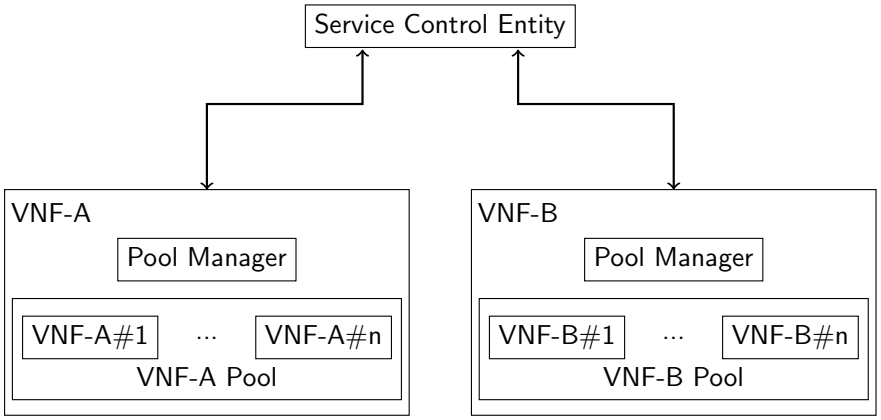

Figure 5.11: VNFPool reference architecture

chain is here defined as an ordered set of service functions that must be applied to packets in a given network service, and according to the VNFPool principles described above, VNFPool is orthogonal and somehow invisible to SFC, even if their concepts are compatible and VNFPool enabled VNFs can be orchestrated by a SFC framework. The current IETF specification for the VNFPool architecture describes the open challenges and issues for the application of reliability and redundancy management to VNFs:

- Redundancy model inside VNF: the selection and placement of backup VNF instances in the physical infrastructure is a key VNFPool aspect and should be dependant on the given type of VNF. Active and backup instances should be placed geographically closed, but most probably not in the same server to avoid outages of backups in case of hardware failures.

- State synchronization inside VNF: service states specific for each type of VNF (e.g. NAT translation table, TCP connection states, etc.) need to be synchronized among the live VNF and its backup instances. The Pool Manager should be responsible to coordinate these mechanisms inside each VNFPool, as well as to maintain some pool states related to redundancy settings, backups locations, etc.

- Interaction between VNF and Service Control Entity: the communication between the Pool Manager and the Service Control Entity should be as more transparent as possible with respect to the capabilities and characteristics of the VNFPool. However, some information exchange is needed for VNFPool addressing, and above to inform the Service Control Entity of any change in VNF capabilities (VM capacity, bandwidth, processing constraints, etc.) when moving from a live instance to a backup in the pool.

- Reliable transport: reliable transport protocols should be considered for the delivery of VNFPool control messages, e.g. for redundancy management, selection of active/backup VNFs, etc. MPTCP and SCTP could be good candidate and should be evaluated for their applicability in VNFPool.

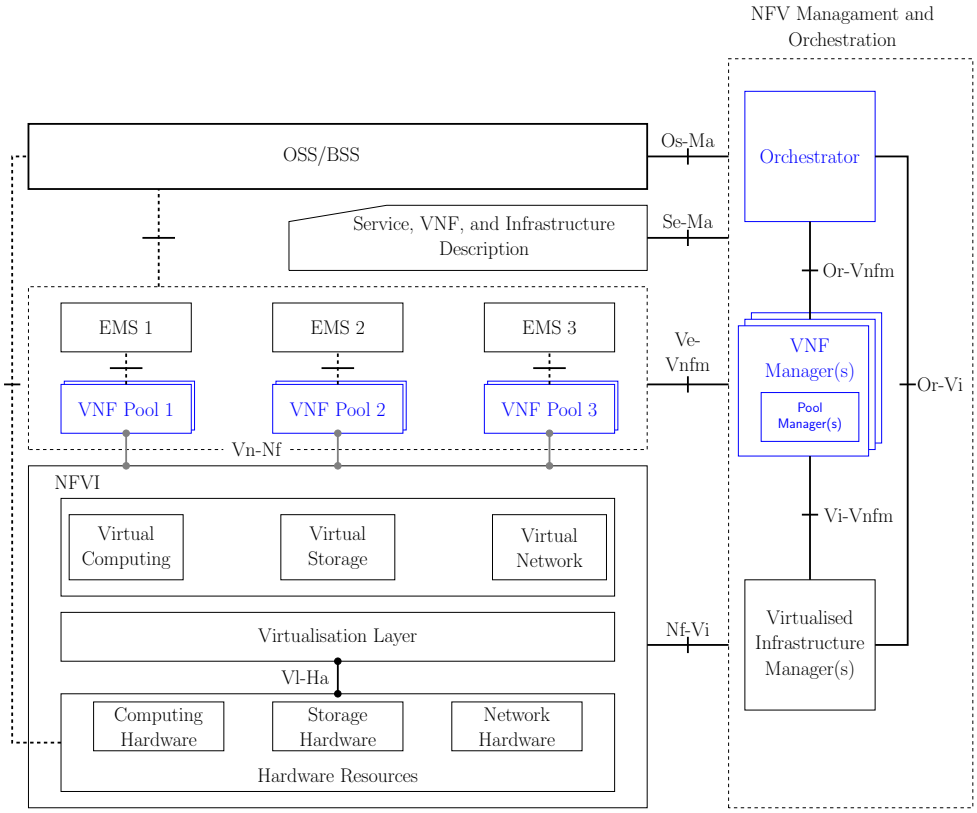

Figure 5.12: VNFPOOL-enabled ETSI NFV reference architecture

5.3.2.3 VNFPool in the Trilogy 2 /NFV architecture

The VNFPool concepts introduced above perfectly fit in the Trilogy 2 architecture, and contribute to fill at least the gap of VNF resiliency with respect to the initial draft of the Liquid Net architecture. Indeed, Trilogy 2 has included the ETSI NFV concepts as a particularisation of its architecture, and VNFPool aims to provide those resiliency and VNF pooling mechanisms not initially considered in ETSI. Figure 5.12 shows that the VNFPool concepts and mechanisms fit in the NFV management and orchestration layer of the ETSI NFV ISG architecture. The VNF Manager is responsible for the management of VNF lifecycle, and provides control functions for instantiation, update, query, termination of VNFs. Different deployment models are supported in this NFV architecture for VNF Managers: one per VNF, or a single one for multiple VNFs.

Most of the VNFPool resiliency mechanisms described above can be implemented as VNF Manager dedicated control functions: indeed the Pool Manager in Figure 5.12 can be included in the ETSI NFV architecture as a component of the VNF Manager, responsible for managing pools of VNFs providing the same network function and supporting both resiliency and VNF scaling (upgrade/downgrade) functions. On top of the VNF Manager, the NFV orchestrator, which is in charge of the orchestration and management of the NFV infrastructure and virtual resources also providing primitives for VNFs chaining, matches the Service Control Entity in the VNFPool architecture (as shown in Figure 5.12).

5.3.3 IRTF: NVFRG

A new RG for NFV specific issues was proposed to the IRTF for the IETF'90. It has met in the IETF'90 and IETF'91 meetings. The group is currently co-chaired by Diego López, from Telefónica I+D, who is an active member of the Trilogy 2 project.

Areas of interest

The NFVRG lists following areas of interest in their WIKI [12]:

- Network and service function chaining: architecture and implementation (e.g. automation of service chain building)
- Autonomous service orchestration and optimization
- New operational aspects of network and service virtualization, as well as new operational models required by virtualization
- Infrastructure and service function description and programming (languages, APIs, frameworks for combined processing, network and storage programming)
- Virtualized network economics and business modeling
- Security, trust and service verification
- Real-time big data analytics and data-centric management of virtualized infrastructure
- New application domains enabled by virtualized infrastructure and services
- System wide optimization of compute, storage, network and energy efficiency
- Explore infrastructure and service abstractions enabled by virtualization
- Autonomic and real-time orchestration enabled by network function virtualization

Relevance to Trilogy 2

Most of the research topics targeted by the NFVRG are relevant for the architecture discussions in Trilogy 2 and beyond.

5.4 Conclusion

The ETSI Network Function Virtualisation Industry Study Group architecture presented in Deliverable D2.1 has been the basis of our work during this period and we have been able to identify different gaps when handling liquidity. The additions we present here it give us the tools to overcome them. We will continue with the development of the OAM framework, as it gives us a reliable tool to check that the NFVs within a chain work correctly. With regards to the pooling mechanisms, we will endeavour to find a fitting venue to this work as it is key for achieving operator grade reliability with software-based, virtualised components.

6 Conclusion and Next steps

At this point, we have a very consistent framework that covers most of the aspects of Network Liquidity we set forth to explore in Trilogy 2. This architecture will be tested within the project, mainly in WP3 and will be the base for further standardisation work.

6.1 Standardisation

The architecture presented at and adopted by the ETSI Network Function Virtualisation Industry Study Group, which is the base of this and the previous architecture deliverable of Trilogy 2, is serving as the foundation for the second phase in the work of this Industry Study Group in the ETSI, which is due to start during the last year of Trilogy 2. Additionally, we have also different WGs at the IETF and the IRTF to standardise our work. Any further findings stemming from the implementation of the use cases will be fed into the adequate SDO.

6.2 Further work

As stated above, the work now will go on testing the architecture against the use cases in work package 3. Additionally, we will continue defining the information models for the different components. We are releasing a first set of tools for controlling liquidity in parallel with this deliverable (see Deliverable D2.4). This set of tools will be checked and refined during the next phase of the project and will result in the final set of tools for controlling liquidity of Deliverable D2.5. This final set of tools will also include information models that take into account all aspects of liquidity developed in Trilogy 2.

Bibliography

[1] D7 overall architecture including design principles. Deliverable 7, Trilogy Project EU 7th Framework Project ICT-216372, jun 2009.

[2] Sam Aldrin, Ram Krishnan, Nobo Akiya, Carlos Pignataro, and Anoop Ghanwani. Service Function Chaining Operation, Administration and Maintenance Framework. Internet-Draft draft-aldrin-sfc-oam-framework-01, IETF Secretariat, October 2014. I-D Exists.

[3] L. Andersson, H. van Helvoort, R. Bonica, D. Romascanu, and S. Mansfield. Guidelines for the Use of the "OAM" Acronym in the IETF. Technical Report 6291, IETF Secretariat, June 2011.

[4] Marcelo Bagnulo, Christoph Paasch, Fernando Gont, Olivier Bonaventure, and Costin Raiciu. Analysis of MPTCP residual threats and possible fixes. Internet-Draft draft-ietf-mptcp-attacks-02, IETF Secretariat, July 2014. IESG Evaluation.

[5] Alistair Barros, Marlon Dumas, and Phillipa Oaks. Standards for web service choreography and orchestration: Status and perspectives. In *in Proceedings of the Workshop on Web Services Choreography and Orchestration for Business Process Management*, 2005.

[6] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. RFC 3135: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, June 2001.

[7] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: Defining tomorrow's internet. *IEEE/ACM Trans. Netw.*, 13(3):462–475, June 2005.

[8] Thomas Erl. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

[9] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. Technical Report 6824, IETF Secretariat, January 2013.

[10] Fernando Gont. Security Assessment of the Transmission Control Protocol (TCP). Internet-Draft draft-gont-tcp-security-00, IETF Secretariat, February 2009.

[11] Jim Guichard and Thomas Narten. Service function chaining charter. Technical report, IETF, Dec 2013.

[12] Jim Guichard and Thomas Narten. Network function virtualization research group (nfvrg) (proposed). `http://trac.tools.ietf.org/group/irtf/trac/wiki/nfvrg`, Dec 2014.

[13] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend tcp? In *Proc. ACM IMC*, 2011.

[14] KantaraInitiative. `https://kantarainitiative.org/`. note.

[15] Daniel King, Marco Liebsch, Peter Willis, and Jeong dong Ryoo. Virtualisation of Mobile Core Network Use Case. Internet-Draft draft-king-vnfpool-mobile-use-case-01, IETF Secretariat, June 2014.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems, August 2000*, 2000.

[17] Arne Koschel, Sabina Hofmann, and Irina Astrova. *Standardization in Cloud Computing*, pages 195–204. WSEAS Press.

[18] Ram (Ramki) Krishnan, Anoop Ghanwani, Pedro Gutierrez, Diego Lopez, Joel Halpern, Sriganesh Kini, and Andy Reid. SFC OAM Requirements and Framework. Internet-Draft draft-krishnan-sfc-oam-req-framework-00, IETF Secretariat, July 2014. I-D Exists.

[19] Lucian Popa, Ali Ghodsi, and Ion Stoica. Http as the narrow waist of the future internet. In *Hotnets*, 2010.

[20] K. Ramakrishnan, S. Floyd, and D. Black. RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP , September 2001.

[21] Andy B. Reid. Architectural Comments on GS NFV MAN001, feb 2014. Contribution to ESI NFV ISG, NFV(14)000014.

[22] Randall Stewart. RFC 4960: Stream Control Transmission Protocol, September 2007.

[23] Radu Stoenescu, Matei Popovici, Lorina Negreanu, and Costin Raiciu. Symnet: Static checking for stateful networks. In *HotMiddlebox*, 2013.

[24] J. Touch, A. Mankin, and R. Bonica. The TCP Authentication Option. Technical Report 5925, IETF Secretariat, June 2010.

[25] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, 2011.

[26] Liang Xia, Qin Wu, Daniel King, Hidetoshi Yokota, and Naseem Khan. Requirements and Use Cases for Virtual Network Functions. Internet-Draft draft-xia-vnfpool-use-cases-02, IETF Secretariat, November 2014. I-D Exists.

