ICT-317756

# TRILOGY2

**Trilogy2: Building the Liquid Net**

Specific Targeted Research Project

FP7 ICT Objective 1.1 – The Network of the Future

# D3.1 Use Case Selection

Due date of deliverable: 30 June 2014

Actual submission date: 31 July 2014

| | |
|---|---|
| Start date of project | 1 January 2012 |
| Duration | 36 months |
| Lead contractor for this deliverable | Université Catholique de Louvain(UCL-BE) |
| Version | v1.0, July 31, 2014 |
| Confidentiality status | Public |

**Abstract**

This document describes a set of use-cases that will be developed within the Trilogy2 project. They allow to achieve previously impossible (or hard to achieve) functions by leveraging the software platforms and cross-liquidity tools as well as their controllers, described in the previous deliverables.
The use-cases are split in three categories, namely mobility, operator infrastructure and wide-area Internet services.

**Target Audience**

The target audience for this document is the networking research and development community, particularly those with an interest in Future Internet technologies and architectures. The material should be accessible to any reader with a background in network architectures, including mobile, wireless, service operator and data centre networks.

**Impressum**

| | |
|---|---|
| Full project title | TRILOGY2: Building the Liquid Net |
| Title of the workpackage | D3.1 Use Case Selection |
| Editor | Christoph Paasch, UCL-BE |
| Project Co-ordinator | Marcelo Bagnulo Braun, UC3M |
| **Copyright notice** | © 2014 Participants in project TRILOGY2 |

# Executive Summary

This document describes use-cases that will be developed within the Trilogy2 project. The use-cases benefit from the liquidity provided by the software platforms, cross-liquidity tools as well as their controllers described in the previous deliverables. The use-cases offer new user-experiences that would be difficult (if not impossible) to achieve without the provided liquidity tools. The use-cases are split in three categories, namely *Mobile Devices*, *Operator Infrastructure* and *Wide Area*.

The use-cases for *Liquidity in Mobile Devices* make extensive use of Multipath TCP to provide a better (or more secure) user-experience. Virtual Machine migration combined with Multipath TCP allows for a seamless application migration, without disrupting a streaming application. WiFi Channel Switching enables resource-pooling from multiple access points in the most efficient way by using Multipath TCP. This allows to provide an improved user-experience thanks to the additional capacity of the different access points. Finally, another aspect of resource pooling uses the fact that one can associate different trust-levels to the different interfaces of a mobile device. Indeed, an open WiFi network provides less security than a 3G/4G connection. Multipath TCP allows to transmit the data in such a way that it becomes more difficult for an attacker to observe security or privacy-sensitive data.

Using our tools to provide *Liquidity in Operator Infrastructure* enables several use-cases that benefit both, operators and their customers. The presented use-cases extensively use virtualisation to benefit from the flexibility of Network Function Virtualisation (NFV). Liquid MPLS VPNs allows a more flexible network architecture for operators offering VPN services to their customers. Similar benefits can be provided to IMS virtualised network functions as the function can easily be placed in data centres in such a way that the resources are used in the most efficient way. Content deliver (e.g., video) with Minicache, a lightweight and high-performing specialized virtual machine, would allow for smaller players in the content-delivery business to enter the market as well, without the need to install content caches (and the accompanying hardware) at the providers.

Finally, *Liquidity in the Wide Area* describes two use-cases that allow data replication across the wide area. In particular, Irminsule enables the propagation of security and hot-fixes to a large number of users. It sacrifices storage to allow the system to benefit from a reduced latency. Further, Cloud Liquidity across the wide area requires the replication of the VM. Block replication allows to migrate an actively running virtual machine while reducing the amount of data that needs to be transferred. This enables users to more easily migrate their VM from one cloud provider to another.

The presented use-cases in this document will be further developed as part of WP3. Namely, the development in deliverable 3.2 and finally the deployment in deliverable 3.3.

# List of Authors

| | |
|---|---|
| Authors | Pedro Aranda, Marcelo Bagnulo, Giacomo Bernini, Olivier Bonaventure, Julian Chesterfield, Jaime Garcia-Reinoso, Felipe Huici, Simon Kuenzer, Anil Madhavapeddy, George Milescu, Catalinx Moraru, Dragoș Niculescu, Christoph Paasch, Costin Raiciu, John Thomson, Ivan Vidal |
| Participants | Intel, NEC, NXW, OnApp, TID, UC3M, UCAM, UCL-BE, UPB |
| Work-package | WP3 |
| Security | Public (PU) |
| Nature | R |
| Version | 1.0 |
| Total number of pages | 35 |

# Contents

# List of Figures

# 1 Introduction

The deliverables from WP 1 and 2 have described software platforms and tools that enable cross-liquidity as well as the control over these. These enable the implementation of new use-cases that are not possible in today's Internet architecture. In particular, *mobile devices* benefit from the resource pooling capabilities and the increased resilience to failures provided by Multipath TCP. In the *operator infrastructure*, Network Function Virtualisation as well as the usage of lightweight, flexible virtual machines enables the operators to scale more easily their infrastructure. Storage liquidity, provided by tools like Irminsule, enable new use-cases and improved user experience in the *wide area data replication*.

In this deliverable we describe several use-cases among the above mentioned three categories. These use-cases will be further developed as part of work-package 3.

## 1.1 Structure of the document

This document is organized as follows.

In Section 2 the chosen use-cases for Mobile Devices are explained. They all make extensive use of Multipath TCP as it provides the required flexibility. Section 3 provide details for use-cases in the operator's infrastructure by using Network Function Virtualisation (NFV). Finally, Section 4 uses Irminsule to trade storage capacity for a reduced latency to quickly provide security and hot-fixes to a large number of end-users. It also introduces wide area block-migration to efficiently move virtual machines across the Internet.

# 2 Liquidity in Mobile Devices

## 2.1 Migrating virtual machines

### 2.1.1 Use-case description

The current use-case presents a scenario where a live streaming app (e.g.: Youtube App) is moved between devices. This allows the user to take advantage of the resources available in the nearby devices without affecting the app user experience. The interaction between the user and the application is continuous, making the device-to-device transition as transparent as possible.

Currently when installing a software application on a device, the application is tightly coupled to the device and cannot be easily moved to different devices. Moreover, moving a live application (i.e., moving an application while it is being executed) to other devices presents a set of challenges and is usually done in data centres for adjusting hardware load. Even in this case, moving a live application has a set of limitations (e.g. network resources cannot be easily moved outside of the local network).

The challenges highlighted in this use-case are related to maintaining the network connections of the application open during the process of moving the application between devices. While it is possible to fully move the state of an application from one device to another, keeping the network connection active is proven to be difficult.

Currently, users who would want to switch interaction from a mobile device to a desktop computer, while watching a live stream, would have to open the application on the 2nd device, and configure it to play the stream. By doing this, the overall user experience of watching a video stream is degraded, by requiring the user to perform a set of specific actions for starting a second instance of the player and connecting to the video stream. These actions take time to be completed and might cause the user to miss important parts of the stream. The solution we propose is to move the application itself keeping all the application state including the playback buffer and the current playback position. On the destination device, the application continues rendering the video stream without requiring any additional steps.

This use-case covers the migration of applications between devices with a different display form factor or resolution. A typical scenario is to push a live car racing broadcast from a mobile phone with a tiny screen to a desktop computer (See figure 2.1). This scenario liquidizes bandwidth and CPU resources between the

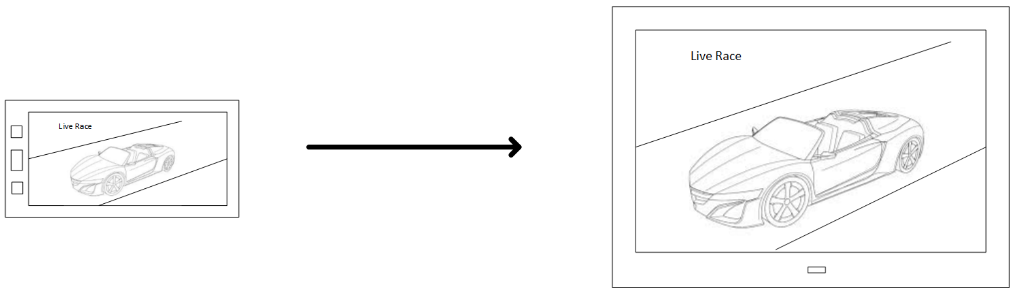

Figure 2.1: VMMig - Live racing match migration from a phone to a tablet

source and the destination devices. It would also allow to liquidize energy by pushing a CPU-intensive task to a different device with better computing capabilities (e.g. hardware video decoding) that require less energy for performing the same actions.

### 2.1.2 Liquidity tools

This use-case maps naturally on the Virtual Machine Migration for Mobile Devices (vM3) liquidity tool.

The use-case is built around moving a live streaming video application across two devices. For the application to be easily moved from one hardware platform to another one, it is encapsulated in a thin virtualisation layer. Thus, moving the application is translated to a virtual machine migration. Recent advances in the hardware platforms for mobile devices [1] [2] allow the usage of virtualisation to be competitive from a performance and energy consumption perspective.

Because the use-case centres on a live streaming application, the migration process must focus on providing a smooth and rapid transition between devices. This is accomplished by liquidizing the network resource of the devices. The MPTCP protocol is used inside the virtual machines to allow network changes without breaking application sockets. Moving apps between devices with different subnets causes the network IP address to change. By using the Break-before-Make capability of MPTCP the connections remain active during the VM transition.

Another important aspect of making the use-case work is having the capability to run the same application, in our case a live streaming application, on different devices. Those devices may differ in terms of hardware capabilities like display size or sensor access up to the operating system and architecture ABI. The solution to those constraints is using the Virtual Machine Migration (vM3) solution.

However, Virtual Machines are not currently deployed on near-user devices ranging from mobile phones and tables to laptops and desktops. The tool handles those issues in two ways: by taking advantage of hardware acceleration and by providing a custom designed virtual machine.

KVM is an open-source virtualisation solution that converts the Linux kernel into a hypervisor, taking advantage of existing hardware virtualisation technologies like Intel VT-x. Migrating a VM means moving its context (memory, disk and network) from one hypervisor to another. Using the vM3 liquidity tool this process happens without service interruption or network loss.

The virtual machine container is filled with only the minimal software requirements for the respective application to run. For our use-case Yocto is used to generate the small footprint Linux image with an X server and an HTML5 capable Webcore. Yocto is highly configurable and can mix only the necessary drivers and packages needed for the architecture.

The video stream is played by an HTML5 video player application. This application runs in the context of a stripped down Chrome webcore. Having an HTML5 webcore comes with the advantages of using HTML5

[1] Asus Zenfone with Intel Atom Z2520:
http://www.cnet.com/uk/news/asus-launches-the-zenfone-series-in-southeast-asia/

[2] Intel Atom Z2520i: http://ark.intel.com/products/75203/Intel-Atom-Processor-Z2520-1MB-Cache-1_i20-GHz?wapkw=intel+atom+z2520

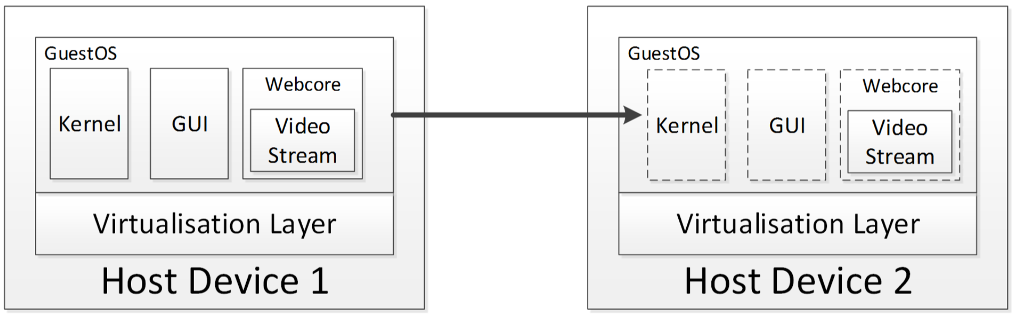

Figure 2.2: Live video streaming migration using the vM3 liquidity tool

Form factor with CSS3 mechanisms. The most important advantage is automatic interface scaling to match different screen resolutions, resolving the problem of having adaptive GUI.

Figure 2.2 shows how the use-case implementation is mapped on the Virtual Machine Migration for Mobile (vM3) devices architecture.

This use-case is made possible thanks to a unique combination of several existing technologies. We are using existing Virtual Machine migration concepts along with MPTCP to enable transfer of active connections across networks. We bring all of this close to the user by taking advantage of existing mobile virtualisation capabilities like VT-x and also distribution configuration projects like Yocto.

Other possible solutions for implementing this use-case rely on stopping the stream at the source and restarting it at the destination. However those solutions are not as good as vM3. Stopping and restarting the stream induces a break in the viewing process that can be crucial to the user. Our implementation ensures that the video stream remains active on the source during the migration process. This actually means that the user keeps interacting with the application until the migration process is completed preventing poor network performance from degrading user experience. Even in the case of high-bandwidth network connections between the source and the destination devices the application continues to run on the source device until the migration process is completed. This ensures a seamless transition between devices.

### 2.1.3 Use-case's interest

This use-case combines two possible liquidity types.

The first type involves network liquidity. This means making the available bandwidth accessible to applications running on a different device. A quick example would be moving the video stream application from a tablet having a poor WiFi connection to a mobile phone with a better 3G connection (or vice-versa).

The second liquidity type refers to CPU liquidity. Using the previous example the video stream can be moved from a mobile phone performing multiple simultaneous tasks to an idle tablet that has enough resources to render the stream at a high quality.

The video streaming use-case impacts the end-user directly by being applied to the devices a regular user interacts with every day. The benefits of liquidizing CPU and network are therefore perceived by the user in the form of better UX and less constrains related to moving apps from the device where they were installed.

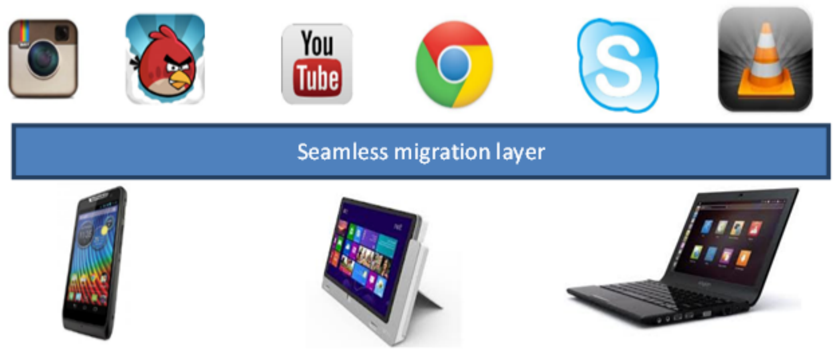

Figure 2.3: Abstraction layer between various devices and various apps

This use-case opens the possibility to move from liquidizing the hardware resources, to liquidizing the applications and removing the boundaries between hardware entities. The migration layer (see figure 2.3) would be a less powerful feature if the apps connectivity would suffer when moving from one device to another.

## 2.2 WiFi Channel Switching

In typical WiFi usage, a client attaches to a single access point (AP) based on a decision that relies on past attachments, and on current AP signal strength. This association is kept as long as the signal stays above a certain limit, regardless of availability of new APs, or of new capacity in neighboring APs. With mobility, the above decision has to be repeated, as the client sees a changing set of APs with different signal strengths, wireless loads, and network loads. This use case explores the possibilities of a client connecting to multiple APs and quickly switching from one AP to another to improve the user experience.

### 2.2.1 Use-case description

All use cases proposed are standard WiFi usage scenarios, but under more challenging conditions:

- *Nomadic usage:* the user is static for minutes or longer, but the environment is highly dynamic. There are many APs, and there is a high churn for user membership on each AP. Traffic is bursty on all channels, on all APs, and from all users. The goal is to improve average user experience, when compared with the standard policy of associating to a single AP.

- *Mobile usage under light load:* the mobile employs association to multiple APs on the same/different channel to overcome L2 handover performance. Radio conditions to the current AP set change permanently.

- *Mobile usage in loaded network:* this is a combination of the previous two so that mobile's own radio conditions change because of mobility, and competing traffic is bursty, which puts stress on adaptation algorithms across the entire stack: association at L1, rate adaptation at L2, windows and retransmissions at L4.

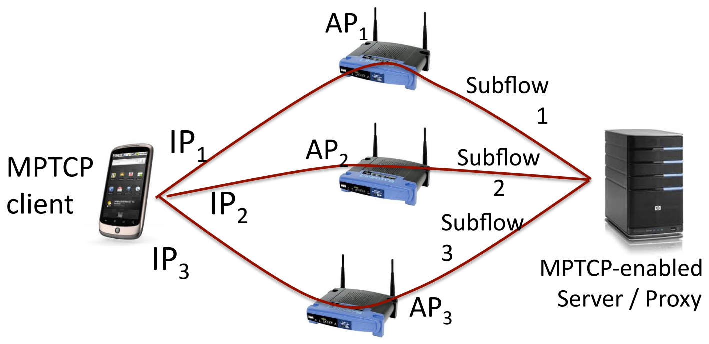

Figure 2.4: Use of MPTCP across multiple WiFi APs.

### 2.2.2 Liquidity tools

There are two techniques considered for this purpose, and both make use of the liquidity provided by MPTCP. One is usage of all APs on a given channel, and the second is switching channels to harvest capacity. As seen in Figure 2.4, in both cases the mobile uses several identities to simultaneously associate to several APs. In fact, it needs a different MAC address for each of its identities, so that it appears as a different mobile associated to each AP. Both mechanisms require usage of MPTCP to associate several IP addresses to the same socket.

### 2.2.3 Use-case's interest

The goal is to improve WiFi user experience in the presence of high population of devices, and high number of APs. The interest for these use cases are driven by the following trends in WiFi usage:

- *growing population of devices.* Even for the nomadic case, there is a high turnover in user population on any given WiFi channel. When other devices are associated to a neighboring AP on the same channel, they will cause a decrease in capacity for the current association.
- *growing number of APs.* In uncoordinated deployments, the user may have a choice between different providers. Current WiFi association policy does not try to associate to other APs when the current one becomes loaded.
- *growing number of channels.* With the advent of the 802.11ac, the 5GHz band channels become popular, therefore more capacity will be available than with the current 2.4GHz band.

In all these cases there are available resources on other APs either on the current channel, or on different channels. Multiple AP association primitive, and the channel switching primitive are the proposed liquidity tools that aim at harvesting this capacity with the help of MPTCP.

## 2.3 Trusted resource pooling with Multipath TCP

### 2.3.1 Use-case description

Smartphones and tablets are becoming one of the most widely used devices to access the Internet. Today's smartphones are equipped with several wireless interfaces (WiFi, 3G/4G, Bluetooth, . . . ). Faced with a huge growth of the data traffic [3, 7], mobile network operators are exploring alternatives to 3G/4G to provide Internet connectivity.

Researchers have explored the interactions between WiFi and 3G in the past. Several studies have demonstrated that there are performance and cost benefits in offloading data traffic to the WiFi network [10]. These findings encouraged network operators to roll out large WiFi networks. For example, FON[3] gathers more than twelve million WiFi access points, most of these being controlled by network operators.

From a pure cost viewpoint, users and network operators could wish to offload their traffic onto WiFi networks. However, WiFi networks also have some drawbacks. First, a WiFi network may be much smaller than 3G, in particular when many users are attached to a low bandwidth broadband link. Second, using WiFi may expose the users to more types of attacks and security problems than 3G/4G networks. Cellular networks are typically controlled by the network operators and it is difficult for attackers to capture or inject packets inside these networks. On the other hand, WiFi started as a completely open technology and there are still many open access points where all data packets can be easily eavesdropped. Users attached to open WiFi networks are vulnerable to a wide range of attacks such as the Firesheep[4] Firefox extension that allows to hijack HTTP sessions. Furthermore, many ADSL/cable routers that often provide WiFi access have often been the target of attacks, some having compromised hundreds of thousands of routers (see e.g. [11, 6]). Once compromised, such a WiFi router can easily mount various types of man in the middle attacks.

Application-level encryption with SSL/TLS is considered by many to be the best option to improve the security of the Internet [9]. However, these encryption techniques are still not deployed everywhere. In 2012, we surveyed the top 10,000 Alexa web sites and found that 38% of them supported HTTPS. A recent survey over the Alexa top 1,000,000 web sites revealed that only 45% of them supported HTTPS [15]. We clearly cannot assume that SSL/TLS will be used everywhere and measurement studies on smartphones show that plaintext protocols continue to dominate [3, 7, 4].

Given that usually the user can trust its mobile network operator (and 3G/4G includes protocols to authenticate the mobile network), in this use-case we propose to distinguish two types of network interfaces :

- **Trusted interface**. An interface is considered to be **trusted** when the user can expect that passive and

[3] `http://www.fon.com`

[4] See `http://codebutler.github.io/firesheep/`

active eavesdropping will be impossible on this interface given its nature (e.g. physical wire) or due to the utilization of encryption techniques.

- **Untrusted interface**. An interface is consider to be **untrusted** if an attacker can easily eavesdrop packets.

We thus suggest to schedule traffic across the different interfaces in such a way that the security-critical or privacy-sensitive data is always sent over the trusted interfaces, while the other traffic can be sent on untrusted interfaces. An attacker would need to be able to eavesdrop the traffic from the trusted interfaces in order to launch an attack. Although, this does not provide a guaranteed security for the traffic, it makes the attacker's life a bit harder.

### 2.3.2 Liquidity tools

Multipath TCP and its Linux Kernel implementation (described in the Deliverable 1.1 on Software Platforms) was designed with resource pooling in mind [17] and aims at distributing data fairly over different interfaces. We use Multipath TCP in order to achieve the above described security goal. Multipath TCP allows to achieve this as it is able to transmit a single data stream via different paths. In the current Internet's protocol-suite, such multipath transmission is only possible with SCTP-CMT [8]. However, widespread deployment has still not been possible with SCTP due to its numerous issues with middlebox traversal.
The ability to steer traffic within Multipath TCP along a specific subflow is possible thanks to the scheduler framework, described in Deliverable 1.3. In our Linux Kernel implementation, we extend the interfaces table in the Linux kernel with one per bit interface that indicates its trust level. This trust level will typically be automatically configured by the application, usually the connection manager, that controls the utilization of the network interfaces. We expect that wired interfaces such as Ethernet could be considered to be trusted by default. However, some companies could prefer to consider that only the company's Ethernet is trusted and rely on 802.1x to verify that the device is attached to the company network. Virtual Private Network solutions built with IPSec, SSL/TLS or DTLS usually provide a virtual network interface. Such interfaces will be considered as trusted by the connection manager. For wireless networks, we expect that 3G and 4G networks will be considered to be trusted given the utilization of link layer encryption to secure the wireless channel. Mobile network operators often install tailored connection managers on the smartphones that they sell. This connection manager could easily recognize the operator's networks and consider them to be trusted. For WiFi networks, the level of trust could depend on the use of link-layer encryption (e.g. WPA2 could be considered trusted while WEP would not be) and also on the utilization of 802.1x (e.g. a corporate WiFi network using EAP-TTLS would be considered to be trusted once the network certificate has been validated).

### 2.3.3 Use-case's interest

The bandwidth liquidity offered by multiple paths between two end-hosts has already been exploited in many ways by using Multipath TCP. This use-case adds another dimension to the network's liquidity by considering the different trust-levels one can have in each path.

Additionally, privacy concerns are rising nowadays. The Internet's users are demanding for more secure and private communication across the Internet. Leveraging Multipath TCP's capability to send traffic across different paths is an easy way to make an attacker's life harder.

# 3 Liquidity in Operator Infrastructure

## 3.1 ClickOS migration

### 3.1.1 Use-case description

Network operators manage their networks by routing traffic via middleboxes——network devices that maintain per flow state (at L4 or application level) and perform application specific optimizations. For instance, cellular operators run HTTP proxies on web traffic, firewall most traffic, shape Skype traffic, and so forth.
Ensuring that the right traffic goes through the appropriate middlebox is not easy. The standard approach of placing all middleboxes "on-path" of the traffic does not scale, because client traffic might be load-balanced via multiple paths to ensure the appropriate amount of bandwidth. Additionally, as traffic is load-balanced across paths, it may be necessary to migrate flows from one path to another, and to move the corresponding state or migrate the whole virtual machine.
The goal of this use case is to enable load balancing of traffic and middleboxes across networks both in the same administrative domain and in the wide-area (across the Internet). We need two techniques to implement such load balancing: **flow migration** and **middlebox migration**.
Flow migration is today possible by using techniques below the IP layer, including MPLS and Openflow. These techniques have a number of disadvantages:

- Both MPLS and Openflow are restricted to a single autonomous network: migration cannot happen across wide-area paths, because MPLS/Openflow are only deployed within single administrative domains.

- Classifying traffic and running it through a chain of middleboxes poses great challenges for Openflow. Openflow offers great flexibility to route flows based on policy implemented by a centralized software controller, but predicting how a packet will be transformed by a middlebox is not easy. The state of the art tries to automatically infer how the packets will be transformed by a middlebox and then create the specific Openflow rules, however the accuracy of this technique is rather poor, only 90% [13]. In contrast, MPLS adds a flow label to each packet that can potentially be used for middlebox chaining.

- MPLS tunnels are widely used in operator networks to enable load balancing. The difficulty with using MPLS to implement this use case is that we need dynamic policies of associating TCP flows with MPLS tunnels, which requires network-wide coordination: the middleboxes classifying the traffic must either be situated at strategic points (e.g. ingress and egress) or they must communicate with the ingress and egress routers in the MPLS part of the network. Additionally, classification may not scale because we may need per-flow entries in the border routers to enable proper load balancing.

Middlebox migration is possible for x86 VMs (e.g. Xen [2]). The key idea of migration is to copy the memory of the VM from one host to the next, pause the VM and resume on the new machine; it is assumed

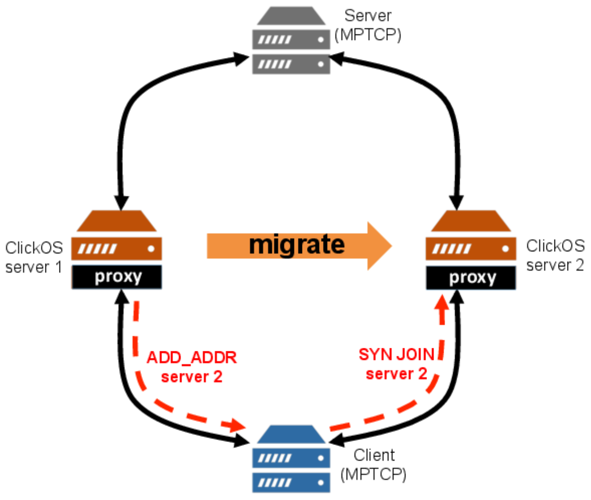

Figure 3.1: Implementing wide area laod balancing with ClickOS and Connection Redirection

that the disk is network mounted, so disk migration is not an issue. However, standard virtual machines use a lot of memory (on the order of hundreds of megabytes or more) and migrating such data takes a lot of time. Further, migrating network connections is a bit trickier: Xen (and other hypervisors) supports it only within the same L2 domain; this is because ongoing connections are maintained by relying on an ARP trick (the migrated VM sends an unsolicited ARP response to inform the switches of its new location). In fact, any wide-area migration technique requires some sort of mobility support which is missing from TCP today.

To implement this use case we will rely on two main liquidity tools developed by Trilogy2: connection redirection and lightweight, application specific virtual machines. The use-case will show how it is possible to perform the following tasks in real time:

- The ability of a network operator to load balance Multipath TCP connections across different wide-area paths by using connection redirection.
- The ability to migrate ClickOS virtual machines between different data centres in much shorter time than the corresponding x86 VM.
- The ability to migrate a ClickOS virtual machine with **per connection state** and its corresponding traffic flows across different data centres.

A high level overview of this use-case is shown in Figure 3.1. The Multipath TCP traffic flowing from the client to the server arriving at ClickOS server 1 is redirected to ClickOS server 2. Before the migration takes place, the ClickOS VM may also be migrated across, depending on whether we are doing VM migration too, or simply traffic load balancing.

### 3.1.2 Liquidity tools

This use-case relies on two key liquidity tools developed by Trilogy 2: connection-redirection and lightweight appliances (i.e. virtual machines), tailored to specific tasks.

Connection redirection allows a middlebox on path of a Multipath TCP connection to redirect via another waypoint(s), by re-utilizing Multipath TCP's in-built mobility mechanisms, namely **the unique connection identifier** allocated by each endpoint. This allows multiple subflows and different addresses to be added to the same Multipath TCP connection. In particular, on path middleboxes can simply spoof ADD_ADDR messages and still sign them with the proper connection key, as long as they have seen the three way handshake. Receiving an ADD_ADDR message tricks the client into opening a new subflow to the specified address, thinking it is reaching the server. The middlebox (ClickOS server 2 in Figure 3.1) can simply proxy the SYN packet received from the client, changing its source address to match its own, and changing the destination address to match the server's. The server will reply with the proper SYN/ACK, which is relayed to the client, which replies with the third ACK and the connection is established. Notice that the middlebox only has to shift source and destination addresses on each packet, which is rather cheap to implement.

Connection redirection allows not only redirecting traffic to one middlebox, but to an arbitrary chain of middleboxes under the same administrative control. Chains can be implemented by simply proxying packets at ClickOS server 2 to reach another middlebox, and only then they will be sent to the server.

Lightweight virtual machines help a great deal with migration. Both ClickOS and Mirage, that are continually being developed in Trilogy 2, offer virtual machines that are tailored to specific applications and have very low memory requirements. For instance, simple ClickOS VMs have a footprint of 5MB only, compared to Linux VMs that take 500MB and more. Such small memory footprint has a direct impact on wide-area virtual machine migration, with a potential to significantly reduce both time and bandwidth costs.

### 3.1.3 Importance of the use-case

The use-case we target cannot be implemented today, and the main reason is the lack of mobility which restricts migration in the wide-area. Multipath TCP enables mobility, and our connection redirection duly implements such migration. Additionally, for practical wide-area VM migration the size of the VM and the time taken for migration has great importance. By reducing the memory requirements more than 100 times, our lightweight virtual machines make wide-area migration not only possible, but practical and enable short-term migration, thus fine-grained load balancing.

The recent trend towards network functions virtualisation aims at deploying clusters of commodity servers in operator networks to deploy middlebox processing for the operator and possibly third parties. However, these deployments are mostly static because of the difficulty of migrating VMs and confined to a single operator domain (because of the limitations of Openflow/MPLS).

Our use-case will show how operators can leverage resources in their networks and in neighbouring ones to cope with temporary high traffic bursts, by renting out spare processing capacity and bandwidth and redirecting some of the traffic via the new virtual machines.

Our use-case allows traffic-only load balancing, as well as processing load balancing, and this can be exploited to adapt the type of resources used in different cases. For instance, consider ISP provider A offers cheap network connectivity and expensive compute cycles; it makes sense to rely on load balancing of traffic alone, and less processing. In contrast, if provider B rents out cheap processing but at higher bandwidth costs, it is worthwhile to compress traffic, thus using the cheaper resource (CPU) to save the ore expensive one (bandwidth) and achieving cross-resource liquidity.

## 3.2 Control Plane for Liquid MPLS VPNs

The telco market is rapidly moving towards an Everything as a Service model, in which Network Functions Virtualisation (NFV, i.e the virtualisation of traditionally in-the-box network functions) is often paired with Software Defined Networking (SDN) tools/technologies and advanced Infrastructure as a Service (IaaS) platforms. Primary area of impact for operators is the network edge, and in particular their data centres and Points of Presence (PoP). Operators are looking at their future data centres and PoPs as more and more dynamic infrastructures, with flexible network architectures and control plane solutions to enable shorter time to market for new functions and services to be offered to the customers. This use case refers to the control plane for a specific reference network operator service, i.e. the MPLS VPN services, that allows to automate the standard MPLS VPNs creation in liquid NFV environments. The reference NFV application of this use case is the virtual CPE, that consists in the execution in virtual environments of those network functions traditionally integrated in hardware gears at customer premises, e.g. for MPLS VPNs, BGP, Firewall, NAT, etc.

### 3.2.1 Use-case description

Two scenarios of the control plane for liquid MPLS VPNs are presented: the liquid MPLS VPNs for virtual CPE in the data centre, used to control and configure liquid VPN services for business customers, and the liquid MPLS VPNs in the PoP, to softwarize and shift to the first PoP of the operator the CPE control plane network functions deployed in customer premises.

#### Liquid MPLS VPNs in the datacenter

A network operator business customer buys IaaS resources (e.g. virtual machines, storage areas, etc.) from the same operator and wants to flexibly extend its VPNs (e.g. that interconnect different customer's sites) to incorporate the new virtual assets running in a data centre X into a private cloud. The customer virtual machines (i.e. the IaaS resources) are automatically allocated and configured by the network operator by means of an SDN-enabled control plane for liquid MPLS VPNs in the data centre X, mainly composed of a Cloud Management System (CMS), like OpenStack, and an SDN controller, like OpenDaylight. This control plane also leverages on OpenVirtual Switch running in the data center commodity servers for virtual networks configuration. An additional virtual machine is indeed allocated for virtual CPE control plane functions, and is configured by the SDN controller (e.g. OpenDaylight) to peer with the MPLS routers in the operator network through a Quagga-based BGP-4 instance and announce the routing information for the customer virtual network (i.e. VRFs, IP addresses, etc.), as depicted in Fig. 3.2. The SDN-enabled configuration also involves the configuration of flows/routing tables in the data centre (virtual) switches for the customer virtual

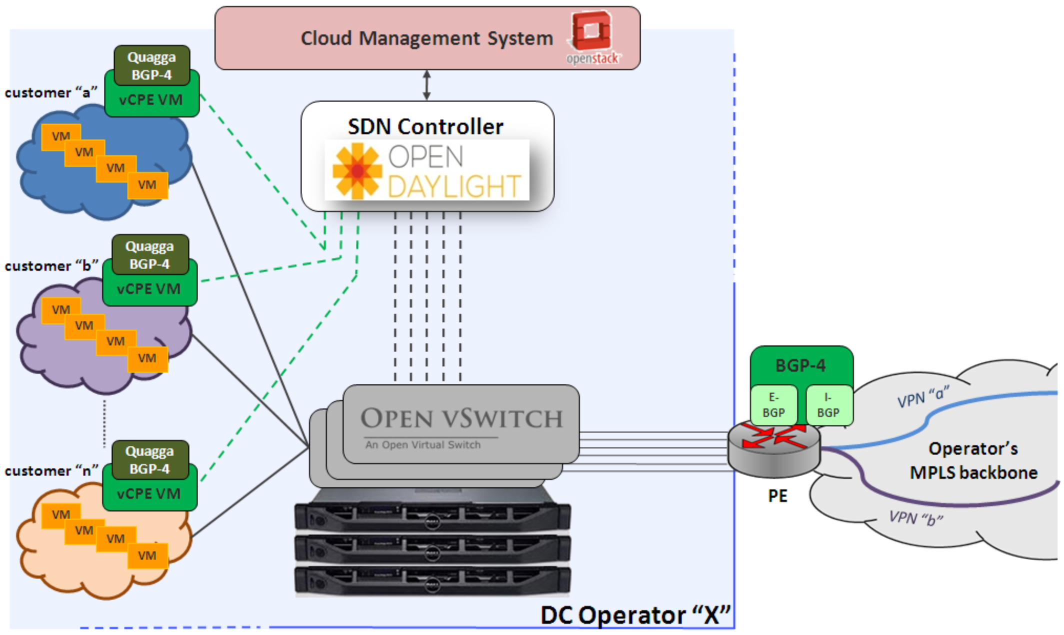

Figure 3.2: Control Plane for Liquid MPLS VPNs in the operator's data centre

machines traffic.

## Liquid MPLS VPNs in the PoP

The complete network configuration needed for the virtual CPE also comprises other elements, like a carrier grade Network Address Translator (CG-NAT), internal DNS servers and other service elements that need to be reachable within the VPN established for a client. All these elements can be supplied as virtual network functions (VNFs) in a data center that is NFV-capable (see Fig. 3.3). From the point of view of the routing information advertised by the different elements, we propose that the CG-NAT advertises the default IPv4 route (0.0.0.0/0) and the service elements are grouped around specific prefixes in the same (private) addressing space used by the end user (e.g. 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16). To do so, we propose to use a virtual machine with a patched Quagga suite that

(i) sends the Layer 3 VPN (L3VPN) routes with the correct *route target* and *route distinguisher* attributes,

(ii) provides a control channel that can be used by a VNF providing MPLS termination and IPv4 routing functions similar to that of a full provider edge (PE) router in order to provide IPv4 connectivity to the CG-NAT or service element VNFs.

This use case can be applied in the case of a movable Home Internet access service: a user has Internet access in his normal residence and moves to his holiday location with the Internet access moving with him. If this move implies a change in the NFV data centre serving the holiday location, it will also imply the move of functionality in order to keep the network overhead at a minimum. When this move implies a significant number of users, it is thinkable that moving the CG-NAT functionality might be beneficial. In this case, as shown in Figure 3.3, the operator can choose to move the control plane, the data plane, or both.

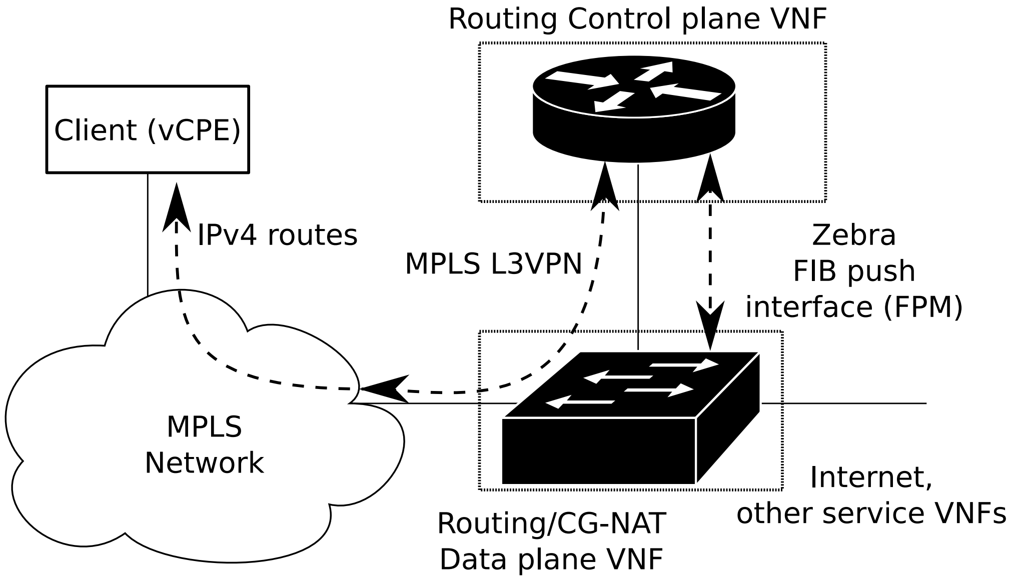

Figure 3.3: Control Plane for Liquid MPLS VPNs in the operator's PoP

### 3.2.2 Liquidity tools

The control plane functionality used in the *Liquid MPLS VPNs* is implemented as a virtual machine running the Linux operating system and a patched Quagga routing suite. It provides the control plane functionality (BPG-4 MPLS VPNs) as well as the data plane control functionality, i.e. the route information that is installed in the VNF implementing the L3 connectivity (CG-NAT, client or data center virtual CPE, routing to other service functions). The data plane control interface is implemented using the Zebra FIB push functionality described in [18].

We propose this interface because it allows to isolate the software license of the Quagga routing software suite, which is the viral GNU Public License (GPL) from the licensing scheme which might be used in the VNF implementing the CG-NAT, the virtual CPE in the data centre, or the virtualised PE router with the service functions.

The benefits obtained by using the proposed tools can be maximized when combining them with virtual machine migration tools.

### 3.2.3 Use-case's interest

The usage of these control plane functions for liquid MPLS VPNs in data centres and PoPs introduces strong innovation in the way operators control and use their virtualised network infrastructure, and above all enables the following benefits:

- Flexible network architectures for operators
- High elasticity to scale up and down (virtualised) resources while optimizing performance

- Higher flexibility to place functions, optimizing network load (liquid "hot spot alleviation") and reducing overhead traffic
- Shorter time to market for new on-demand network functions and services
- Combination of Virtual Machines, SDN tools for more flexible service innovation
- Removal of the Broadband Access Router from the customer premises
- Use of commodity hardware in the PoP and platforms sharing
- Seamless MPLS VPN extensions ("extend my VPN in the Cloud")
- Reduced field operation, maintenance and in-house problem solving for MPLS VPN services
- Pay-as-you-grow + pay-per-use for network and computing functions

## 3.3 Providing resilience to IMS virtualised network functions

### 3.3.1 Use-case description

This use case considers the scenario where a network operator migrates all its IMS infrastructure, consisting of several specialized machines, to a set of virtual machines provided by the fictional *AlwayONvm company*. AlwaysONvm offers a special service for virtual network functions (VNFs) named "Resilient Machines". This service provides the proper procedures to transfer end-users between similar VNFs, which also permits to add resilience after a virtual machine failure. Furthermore, this service is transparent to applications running over compliant end-user devices, so those applications will not be affected after active sessions are moved from old NFVs to new ones. With this service, IMS network functions like x-CSCFs could be deployed over virtual machines as depicted in Fig. 3.4, supported by the tools provided by the NFV architecture.

### 3.3.2 Liquidity tools

A liquid system should allow to maximize the usage of network resources by any application, but this flow of resources has to be controlled, trying to avoid tsunamis affecting end-users. In other words, the instantiation of new resources or after stopping non necessary resources should not affect active sessions established between end-users. In the particular case of IMS visualized network functions, key elements like Proxy-/interrogating-/serving-call session control functions (P-/I-/S-CSCFs), store information of active sessions and users. On the other hand, the IMS stack at the user equipments has state about the network elements (IP addresses and ports) assigned to every session. When moving network resources with liquidity tools like the ones provided by the Network Functions Virtualisation (NFV), it is important to update the proper information, both at network elements and at the end-users terminals too. After an element, described in the NFV architecture, detects an event that requires to add or remove an IMS VNF instance, the corresponding IMS network element in charge of the resilience has to transfer sessions between the affected virtual machines, by using the proper SIP procedures.

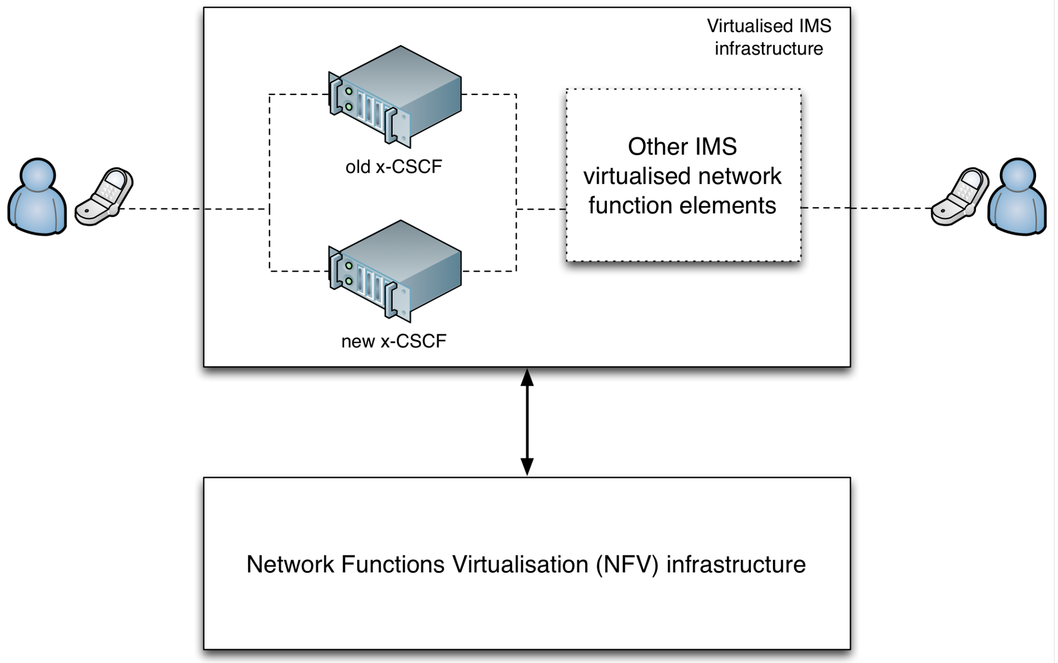
Virtualised IMS infrastructure
old x-CSCF
new x-CSCF
Other IMS virtualised network function elements
Network Functions Virtualisation (NFV) infrastructure

Figure 3.4: Resilient IMS use-case

In the current Internet, such scenario described above would not be feasible to deploy, and a failure of an IMS network function will result in the disruption of all active sessions handled by the failed machine. In a scenario where the network functions are visualized, running over regular commercial-off-the-shelf devices, it is important to take into account resilience. NFV needs to guarantee resilience to be an important tool for the liquid network.

### 3.3.3 Use-case's interest

There are different benefits of using this functionality of resilient IMS visualized network functions:

- It would be possible to transfer users from overloaded virtual machines to existing or recently instantiated machines.
- It would be possible to transfer users from underloaded virtual machines to other already running virtual machines, in order to stop the former to reduce costs/energy.
- It would be possible to transfer users after a virtual machine failure to existing or recently instantiated machine.

## 3.4 Minicache video delivery

### 3.4.1 Use-case description

While the Internet is used to carry packets for a wide range of applications and services, the past years have seen video traffic rapidly expand, to the point where today it comprises the largest portion of all traffic on the network. A recent report states that; consumer Internet video traffic will make up 69% of all consumer

Internet traffic by 2017, up from 57% in 2012 [1]. Within this, real-time entertainment dominates, with Netflix accounting for as much as 33% of peak period downstream traffic in North America [14].

The large majority of this traffic is delivered via Content Delivery Networks (CDNs), that are expected to transport almost two-thirds of all video traffic by 2017 [1]. CDNs optimize video delivery by deploying so-called content caches in different networks, often geographical dispersed, so that user requests can be handled locally by nearby caches. For instance Akamai, the market leader, claims to deliver more than 20% of all Web traffic worldwide [12], and does so by deploying more than 100,000 servers in more than 1,800 locations across nearly 1,000 networks [5].

While Akamai is the largest player many others exist, ranging from traditional CDNs to ISP-operated CDNs, content provider CDNs, free CDNs and even peer-to-peer CDNs [16]. Their business models and infrastructures vary, but what they all have in common is that they would benefit from having a presence, even if small, at a large number of distributed points in networks, such as the Points of Presence (PoPs) that operators deploy at aggregation points in their networks. Proof of this is the fact that Akamai has already started forming alliances with a number of large ISPs, including AT&T, Orange, Swisscom and KT [5].

Beyond such two-way agreements, more recently ISPs have started deploying small data centres called *micro data centres* at PoPs and to allow third parties access to these infrastructures [5]. The availability of such "pay-on-demand", liquid infrastructure would allow CDNs to dynamically expand their capacity and reach, without having to go through the expensive and time-consuming process of deploying hardware and facilities. Such "liquid" video delivery would be especially beneficial to smaller players in the market who cannot afford large up-front investments. To bring this vision to reality we would need the ability to quickly instantiate virtualized content cache close to where content is being requested from. Beyond this, such functionality would allow for quick ramping up of cache instances in order to cope with dynamically increasing load (e.g., flash crowds).

### 3.4.2 Liquidity tools

To achieve the goals stated, we rely on the ClickOS liquidity tool, and in particular its ability to instantiate VMs in a matter of milliseconds while providing the high, 10 Gb/s and higher throughput that video delivery needs; standard virtualisation tools (e.g., a Linux VM running on KVM or Xen) would not be able to achieve such numbers.

ClickOS, however, is meant to mostly run middlebox software, and so several improvements are needed in order for it to act as a content cache. The prototype we are working on is called Minicache, and is based on MiniOS, the minimalistic, para-virtualised OS that ClickOS uses, and the network back-end optimizations implemented in the ClickOS work. In addition to this, Minicache leverages light-weight IP (lwip), a basic IP/TCP network stack included with the MiniOS sources, and adds (1) a fast and simple HTTP server; (2) a hash-based, fast look-up file system called SFS that uses video IDs to map content to block numbers where the data are actually stored; (3) a block cache to optimize access to content and (4) striping and chunking mechanisms so that one video object can be efficiently delivered from several content cache instances.

With all of this, Minicache is able to instantiate content caches in milliseconds and deliver content at rates of several Gigabits per second, thus forming the basis for building “liquid”, virtualised content delivery networks.

### 3.4.3 Use-case’s interest

Video delivery already accounts for the majority of traffic on the Internet, and forecasts state that this trend is only going to continue to grow in the future. In addition, the delivery market is currently dominated by a few players and is likely to remain so given the massive up-front costs related to deploying large numbers of content caches. This use case is aimed at achieving higher video performance delivery while allowing virtual CDN providers, that is, those running over third-party infrastructure such as micro data centres, to compete with the more established players.

# 4 Liquidity in the Wide Area

## 4.1 Hot-fix propagation with Irminsule

Propagating security and hot-fixes to a large number of end-users. Linux distributions and enterprises can use the properties of Irminsule to widely propagate security and hot-fixes to a large number of end-users.

### 4.1.1 Use-case description

Operating System vendors and maintainers output releases periodically. In between standard releases, as is the case with any software development cycles, there may be the requirement to issue a security update or hotfix. Propagating these changes is an asymmetric operation, where end-users can choose when they want to perform a synchronisation. Certain files in an operating system should not change and it would be beneficial if these could be reverted at any point in time back to a good, known state. By using a Git-like system for maintaining a file-system this is made possible.

These fixes when released require pushing to potentially a large number of users within a relatively short time-window. The longer a fix is not applied to a system the more likely it will be compromised or suffer from a known bug.

Irminsule is designed to be the substrate for building self-managing and self-healing systems (described in Deliverable 1.1). It is self-managing in that it has the ability to checkpoint everything in independent branches and lets a scheduler have a lock-free global view of activity that it can use for automatic resource scheduling across diverse system components. The self-healing property arises because the provenance graph allows for the rollback of entire distributed clusters, taking advantage of immutability.

End-users will have a base system that is a replica of the pristine set provided by the provider. The end-user will then change parts of the system, changing configurations and adding new files, drivers, etc. Operating system / software developers' will benefit from providing security updates / hotfixes through a distributed file-system because all changes can be propagated from anywhere in the system where the branch key is equivalent. This reduces the load to the main server and increases the overall bandwidth of the system, akin to BitTorrent type systems.

(i) An Operating system vendor produces a hot-fix that should be applied to customers' machines

(ii) The end user detects that there is a change available through some form of notification system or website

(iii) End user then synchronises the changes with the remote file-system by requesting an update

(iv) If there are no conflicts in the simulation then the changes can be applied to the local system

(v) The system is then synchronised.

### 4.1.2 Liquidity tools

Irminsule is a massively scalable, immutable and branch-consistent storage solution. It uses both storage and bandwidth liquidity.

Distributed databases usually make the distinction between managing concurrency coming from many different users and background administrative tasks trying to asynchronously managing consistency invariants. Each problems have received numerous contributions in the academic and industrial world, and the usual way to solve them is to use "optimistic" techniques, which make the assumption than the need for strong synchronization points on large-scale systems is usually minimal. Irminsule adopts a more generic approach by giving to users and applications the same tools to manage both concurrency and replication. These tools are based on the notion of branch-consistency – dubbed like this as the fork/pull/push/merge life-cycle is similar to the one of distributed version control systems. In this model, application builder have to choose the type of data they want their application to work on and explicit how merge conflicts should be handle. In exchange, they got a loose consistency model, where ACID properties are guaranteed on each branch and where they have a direct control on the replication strategy. Irminsule chooses to use an optimistic replication system because (i) it improves availability: applications make progress even the network is unreliable; (ii) they are flexible because they do not need a-priori knowledge on the underlying network (they can use gossip and epidemic network replications); and (iii) they can scale to billions of nodes because they do not require synchronization. The drawback is that the users have to handle conflicts. In the case of system level file changes, these non-configurations files if managed by a package manager should not be changed and as such the conflicts can all be pre-resolved before reaching the end-users' systems. If there have been changes to the file system then the differences can be compared to the last good known state in the branch and easily reverted before applying any further changes. If there has been a valid reason for a change then these conflicts would need to be handled by an end-user.

The benefit of liquidity in this case is that storage resources are shared, avoiding bottlenecks from any particular resource pool. Storage is sacrificed in order to increase the overall bandwidth of the system, much like seeding in BitTorrent and thereby reducing the latency of all users from receiving the updates.

### 4.1.3 Use-case's interest

This use-case demonstrates one possible solution in the case whereby one particular resource is required by a number of users. Other use-cases are investigating techniques for resource sharing whether it be cross-provider, cross-resource or cross-layer. This use-case demonstrates that by sacrificing one resource - in this case storage - the overall system may benefit in other ways, e.g. overall latency for receiving updates for all users. As resources become more liquid, such tradeoffs will have to be made and this use-case allows for the investigation of how these incentives can be made and shared amongst a group of separate users with a common goal. This particular case is looking at the overall quality of service (QoS) amongst a group of users as traded against localised storage. This use-case also provides some interesting mechanisms that will be needed to promote liquidity in the more general case. The ability to determine if resources have been received by end users and have not been modified in transit will be important. This would necessitate sending some of the meta-information along with the resource in order to validate that the resource is as expected. Irminsule can potentially also be combined with block-level storage replication mechanisms to offer fine grained control of storage resources across the Federated cloud.

## 4.2 Cloud Liquidity

Cloud Liquidity is a potential usage of the storage liquidity mechanism being developed as part of Trilogy 2, wide-area block migration. It allows customers to avoid provider lock-in and have more flexibility and choice when determining where their paid for services are located.

### 4.2.1 Use-case description

Cloud Liquidity helps avoid supplier lock-in. One common concern of many end-users of the Cloud Platform Provider products is that they are tied into a particular providers eco-system and cannot migrate their systems away without significant effort in the export and import stage unless there are specific tools that have been created. These tools are normally created on the import side. Moving towards an open-market with inter-operable standards is being approached in the course of Trilogy 2 but as with all standardisation efforts it will take some time to complete. OnApp, as a Cloud Platform Provider, offers an IaaS platform that many hosting vendors use for creating and managing the end-user customer Virtual Machines. As a step towards allowing complete freedom suggested above, one of the interim measures is to allow customers to migrate their content away from one hosting provider and on to another provider. This facility will allow more flexibility for end-users to choose which cloud that they would like their services to be hosted on. Cloud providers would then need to differentiate their product and service from their competition which will be good for end-users and also as a benefit to providers, open up the market to more potential users if their offering is better. Cloud providers can choose whether to opt in to the federated market and so can choose to restrict their clients from import/export but it is foreseen that opening the cloud to import/export functionality will be a marketable feature that may determine where end-users choose to place their product. It also opens the possibility for mutually beneficial inter-cloud arrangements whereby resources can be located geographically closer to the end-users. Referring to Figure 4.1 it may be the case that a Cloud Provider P1, first established in North America as the client-base was originally located in North America also, represented by C1. As P1 has grown it may find that it has a client base that includes India, represented by C3 and also in Spain, represented by C2. There may be providers that are closer and/or have other beneficial properties such as being cheaper. Provider P2 may be closer and offer a better level of service to C2 and provider P3 could similarly offer a better level of connectivity for client C3. If Virtual Machines can be moved independently then it may be beneficial for the provider P1 to migrate resources either pro-actively to P2 and P3 or for the customers to determine that they may be better served and so decide to migrate the VMs as they choose.

Customers of services such as AWS can migrate snapshots of instances between availability zones but the

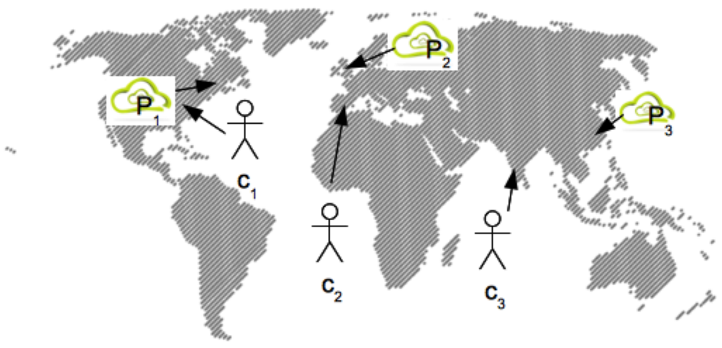

Figure 4.1: Multiple client bases and providers geographically distributed

end-provider of the service is still Amazon. If there is a dispute about the SLA, availability or reliability the end-user would have to choose whether to migrate their resources away. In this use-case scenario the end-user clients have contracts with the individual providers and as such could move between providers. This helps to mitigate against the real issue of provider lock-in and also allows other services to be provisioned such as allowing for the creation of new data-centres and migrating users to the new site.
Possibilities include;

- Planned maintenance windows
- Move workloads dependent on requirements
- Load balancing across data centres
- Cost/Energy savings

### 4.2.2 Liquidity tools

This use-case relies on a new technology that has not yet been presented in Trilogy 2 but also makes use of the Federated Market that has been presented as part of D1.1 - “Software Platforms”. The particular technology that is being developed is wide area block migration. There are file replication mechanisms such as rsync that can be used to replicate files over the wide area network but this differs in that the entire disk content is treated at the file-system level. Block level migration can therefore be used for an actively running system without the need for pausing the VM. Replication with rsync can only guarantee that the VM is completely replicated by suspending the VM for the duration of the replication process. Depending on the VM size and the bandwidth available between replication sites this can take some time. Typical small instance VMs are 10GB. Assuming 50% utilisation, transferring 5GB over a 100Mb/s link would take 7 minutes[1]. Larger instances will take longer. The data centre bandwidth may also be capped at a lower level, increasing the upload time and the time that the VM has to be powered down for. With block level replication there is some

[1] `http://www.zen.co.uk/business/online-data-backup/backup-solutions/data-transfer-times.aspx`

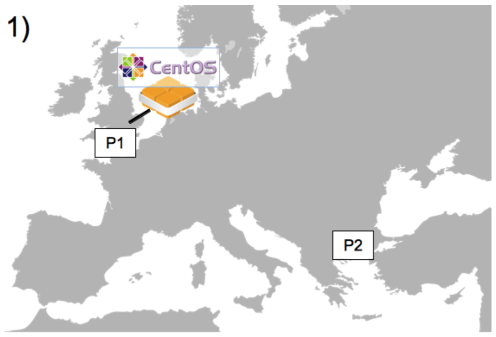

Figure 4.2: Single Cloud running

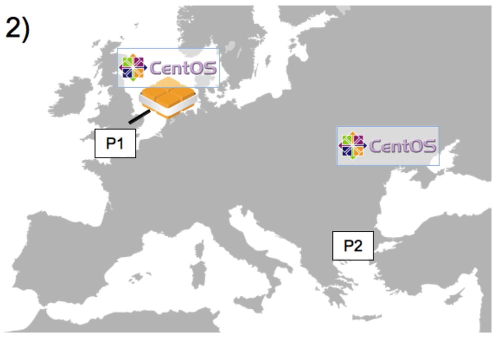

Figure 4.3: Select a site for replication

local buffering to account for writes up to a certain, limit but eventually the bottleneck will be determined by the bandwidth between the replication sites and the write speed will slow down to that of the bandwidth. The wide area block migration mechanism that we are implementing is also sparse-aware in that it can tell which blocks have been written to. This allows for only blocks that have been written to be transferred and translates into a utilisation percentage as mentioned before. For this particular use-case the replication path will not be persistent and as such the cost of maintaining expensive leased lines for replication is not required. Multiple VMs can therefore be migrated using bandwidth only as required.
The process for VM migration is as follows:

(i) The customer or the provider selects a migration target using the OnApp Federated Market (Fig 4.2)

(ii) A target VM is created that matches the source VM type and attributes (Fig 4.3)

(iii) A replication path is enabled between the source and the destination through a button in the UI (Fig 4.4)

(iv) The content replicates in the background. The VM is still usable at this point (Fig 4.4)

(v) The replication finalises and a completion signal is transmitted (Fig 4.4)

(vi) The destination VM is powered on

(vii) The source VM is powered down and depending on the settings, deleted (Fig 4.5)

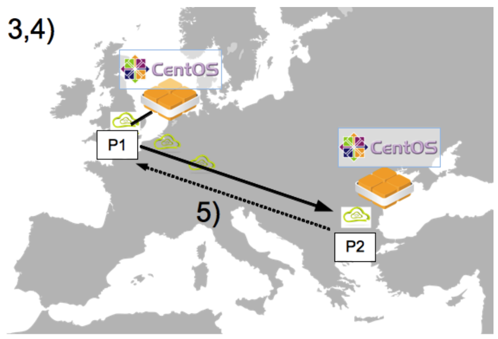

Figure 4.4: Create replication path, replicate and finalise

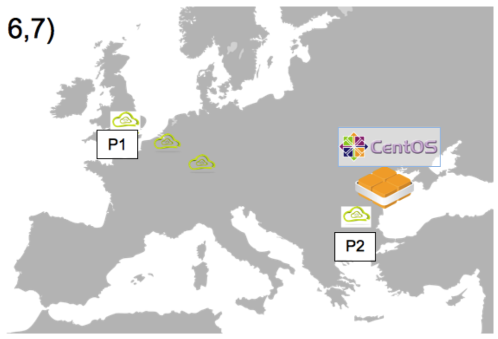

Figure 4.5: Power down source site

### 4.2.3 Use-case's interest

Cloud Liquidity demonstrates many key benefits of storage liquidity and also liquidity in general. By migrating resources across the wide area, end-users can benefit from disparate resource pools that are managed possibly by different providers. The reasons for moving the resources can be explored further in the incentives and enforcement parts of the Trilogy 2 project. Supplier and vendor lock-in is an often cited reason for not investing in 'Cloud' solutions. By giving users the technology to move resources as they require it allows greater flexibility and choice. By opening up the market in this way it will lead to further requirements that the users can demand. One example reason include the recent cases of Governmental access of personal documents through systems such as PRISM. This may lead to end-users wanting greater choice in where to store content.

# 5 Conclusion

To conclude this deliverable, we provide the a summary of the selected use-cases and the tools they require. Some of these tools have not yet been published in a deliverable of this project.

| Use case | Tools |
|---|---|
| Migrating virtual machines | MPTCP (D1.1)<br>Virtual Machine Migration for Mobile Devices (D1.1)<br>Interactions between processing and bandwidth liquidity mechaisms (D1.2) |
| WiFi Channel Switching | MPTCP (D1.1)<br>MPTCP and channel switching interaction (D1.2) |
| Trusted resource pooling with Multipath TCP | MPTCP (D1.1)<br>MPTCP Scheduler infrastructure (D1.3) |
| ClickOS migration | ClickOS<br>MPTCP (D1.1)<br>Waypoint migration: Moving the “middle” of a connection (D2.2) |
| Control Plane for Liquid MPLS VPNs | Virtual CPE (D1.1)<br>Network Function Virtualisation (D2.1) |
| Providing resilience to IMS virtualised network functions | Network Function Virtualisation (D2.1) |
| Minicache video delivery | Minicache |
| Hot-fix propagation with Irminsule | Irminsule (D1.1) |
| Cloud Liquidity | Federated Market (D1.1)<br>Block replication |

Upcoming work will concentrate on implementing and deploying these use-cases in the deliverables 3.2 and 3.3.

# Bibliography

[1] Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2012–2017. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`, 2012.

[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[3] J. Erman and K. Ramakrishnan. Understanding the Super-sized traffic of the Super Bowl. In *ACM IMC*, 2013.

[4] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A First Look at Traffic on Smartphones. In *ACM IMC*, 2010.

[5] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. Pushing cdn-isp collaboration to the limit. *SIGCOMM Comput. Commun. Rev.*, 43(3):34–44, July 2013.

[6] D. Goodin. Guerilla researcher created epic botnet to scan billions of ip addresses. `http://goo.gl/G86ew`, March 2013.

[7] J. Huang, F. Qian, Y. Guo, Y. Zhou, and Q. Xu. An in-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *ACM SIGCOMM*, 2013.

[8] J. Iyengar, P. Amer, and R. Stewart. Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

[9] C. Jackson and A. Barth. Forcehttps: Protecting High-Security Web Sites from Network Attacks. In *WWW*, 2008.

[10] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong. Mobile Data Offloading: How Much Can WiFi Deliver? *IEEE/ACM Transactions on Networking*, 21(2):536–550, 2013.

[11] J. Milliken, V. Selis, and A. Marshall. Detection and analysis of the Chameleon WiFi access point virus. *EURASIP Journal on Information Security*, 2013(1):1–14, 2013.

[12] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.

[13] Qazi, Z. A. and Tu, C-C and Chiang, L. and Miao, R. and Sekar, V. and Yu, M. SIMPLE-fying Middlebox Policy Enforcement using SDN. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 27–38. ACM, 2013.

[14] Sandvine Inc. Sandvine global internet phenomena report. `http://www.sandvine.com/downloads/documents/Phenomena_2H_2012/Sandvine_Global_Internet_Phenomena_Report_2H_2012.pdf`, 2012.

[15] J. Vehent. SSL/TLS analysis of the Internet's top 1,000,000 websites, 2014. `https://jve.linuxwall.info/blog/index.php?post/TLS_Survey`.

[16] Wikipedia. Content Delivery Network. `http://en.wikipedia.org/wiki/Content_delivery_network`, 2013.

[17] D. Wischik, M. Handley, and M. Bagnulo. The resource pooling principle. *SIGCOMM Comput. Commun. Rev.*, 38(5), September 2008.

[18] Zebra FIB push interface. `http://www.nongnu.org/quagga/docs/docs-multi/zebra-FIB-push-interface.html`, aug 2012. note.