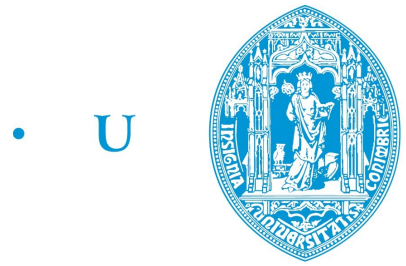
Gestão de Infraestruturas Virtualizadas para Redes de Acesso

By

Nuno Miguel Reis

A master thesis submitted to the University Of Coimbra
for the Master of Science in Computer Science
Communication Networks Engineering

Department of Informatics Engineering
Faculty of Science and Technology
July 2012



Supervisor: Prof. Paulo Simões

Typeset in LaTeX 2_ε.

Acknowledgements

To the virtuoso project team for all the support, for the learn experience and for driving me successfully towards my goals.

To my family, for caring and for being there for me, always.
To my wife Catarina for everything.

Finally I would like to dedicate this to my father who would certainly be very proud and happy to see this day.

Abstract

The way new services and platforms currently offered on the Internet have had a strong evolution in the last few years. The reason behind that is much due to the maturity reached by several virtualization frameworks and because broadband network access is becoming predominant.

Telecom operators are one of the strongest drivers behind these developments which in a way is making virtualization visible in several parts of their networks, from the core parts to data centers and private clouds.

All these developments allow us to estimate that one of the next steps for service providers may involve the virtualization of a strategic element in broadband access networks, the Residential Gateway (RGW). This network element (NE) is currently located at the customer premises but is installed in a very sensitive point between the access network and home network which also makes it a single point of failure in service delivery.

The RGW is a key element in triple play service offers (IPTV, VoIP and Internet), but constitutes a considerable burden for operators in terms of acquisition, operation and maintenance.

This thesis research proposes a full-fledged architecture for virtualized Residential Gateways (vRGWs) replacing the physical device by its virtual counterpart. The virtual residential gateway (vRGW) comes into existence as a virtual entity, a *Platform as a Service* (PaaS) in a private cloud model.

It seems to us that with the adoption of vRGW, it will be possible to significantly reduce costs while introducing important improvements into management and service delivery.

Keywords

Broadband Access Networks, Cloud Computing, Home Networks, Management, Residential Gateway, Virtualization.

Contents

Acknowledgements iii

Abstract v

List of Figures ix

List of Tables xi

List of Acronyms xiii

1 Introduction 1
- 1.1 Research Scope . . . 1
- 1.2 Context . . . 2
- 1.3 Structure of this document . . . 3

2 State of the Art 5
- 2.1 The Case for Virtualized RGWs . . . 6
- 2.2 Management of Virtualized Gome Gateways . . . 8
  - 2.2.1 CPE WAN Management Protocol . . . 9
  - 2.2.2 Extensible CWMP . . . 10
- 2.3 Significance of vRGW Management . . . 11
- 2.4 Related Work . . . 13

3 Methodology and Proposed Approach 15
- 3.1 Proposed Network Architecture . . . 16
  - 3.1.1 Broadband Forum VLAN Aggregation Topologies . . . 16
  - 3.1.2 Proposed Architecture for Support of vRGW . . . 19
- 3.2 Proposed Integrated Management Framework . . . 21

4 Validation 25
- 4.1 An Architecture for Virtualized Residential Gateways . . . 26
  - 4.1.1 Reference Testbed . . . 26
  - 4.1.2 Virtualization of the RGW . . . 27
  - 4.1.3 Experimental Results . . . 28
- 4.2 A Management Framework for Virtualized Home Gateways . . . 31

4.2.1 Experimental Testbed 31
4.2.2 CWMP Proxy Concept 31
4.2.3 Experimental Results 33

5 Evolution of the Research **39**
5.1 Introduction 40
5.2 Publications 41
5.3 Planning 42
5.4 Evolution of the Research 44

6 Conclusion and Future Work **47**

A Publications **49**
A.1 An Architecture for Virtualized Home Gateways 50
A.2 Management of Virtualized Home Gateways 59

B RGW Virtualization Testbed Setup **67**

References **75**

List of Figures

2.1 Classic RGW vs. Virtualized RGW. 7

3.1 VLAN per service model (adapted from [1]). 17
3.2 VLAN per subscriber model (adapted from [1]). 18
3.3 vRGW-enabling network architecture. 20
3.4 Vertical Segmentation vs. Hybrid Segmentation. 21
3.5 CWMP based Integrated Management Architecture. 22

4.1 RGW virtualization testbed. 26
4.2 Measured Throughput per vRGW ("Average User"). 28
4.3 Measured Throughput per vRGW ("High User"). 29
4.4 CPU used by the Hypervisor ("High User"). 30
4.5 Average CPU used per vRGW ("High User"). 30
4.6 CWMP based vRGW management testbed. 32
4.7 1x vRGW (single) *vs.* the mean of 30x vRGW (bulk). 34
4.8 30x vRGW (bulk) cumulative response time. 35
4.9 Registered packet volume on a CWMP request. 35
4.10 Registered byte volume on a CWMP request. 36

List of Tables

4.1 Downstream bitrates for all-digital triple play. 27
4.2 Execution Response Times Descriptive Statistics. 34

List of Acronyms

ACK Acknowledge

ACS Auto Configuration Server

ALG Application Level Gateway

API Application Programming Interface

AS Application Server

BNG Broadband Network Gateway

C-VID Customer VLAN ID

C-VLAN Customer VLAN

CAPEX Capital Expenditure

CNSM Conference on Network and Service Management

CPE Customer Premises Equipment

CPU Central Processing Unit

CWMP CPE Wan Management Protocol

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

DSL Digital Subscriber Line

FEC Forward Error Correction

FTTx Fiber To The Premises

GB Gigabyte

GPON Gigabit Passive Optical Network

HDTV High Definition TV

HGI Home Gateway Initiative

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

IPTV Internet Protocol Television

ITU International Telecommunication Union

JVM Java Virtual Machine

KVM Kernel-based Virtual Machine

LAN Local Area Network

LXC Linux Container

M-VLAN Multicast VLAN

MB Megabyte

MPEG Moving Picture Experts Group

MPLS Multiprotocol Label Switching

MTU Maximum Transmission Unit

NAT Network Address Translation

NE Network Element

NIC Network Interface Controller

NMS Network Management System

OLT Optical Line Termination

ONT Optical Network Unit

OPEX Operational Expenditure

OS Operating System

OSGi Open Services Gateway Initiative

PW Pseudowire

RAM Random Access Memory

RGW Residential Gateway

RPC Remote Procedure Call

S-VID Service VLAN ID

S-VLAN Service VLAN

SDK Software Development Kit

SDTV Standard Definition TV

SIP Session Initiation Protocol

SOAP Simple Object Access Protocol

SOHO Small Office, Home Office

SSL Secure Sockets Layer

STB Set Top Box

TCP Transmission Control Protocol

TLS Transport Layer Security

TR Technical Report

TV Television

UDP User Datagram Protocol

VLAN Virtual Local Area Network

VM Virtual Machine

VoIP Voice over IP

WAN Wide Area Network

X-CWMP Extensible CWMP

XEN Virtualization Hypervisor

XML Extensible Markup Language

vRGW virtual Residential Gateway

1 Introduction

1.1 Research Scope

The research work discussed in this document fills in the scope of the VIRTUOSO Project developed by the Department of Informatics Engineering from the Faculty of Science and Technology in University of Coimbra and partially funded by PT Inovação.

1.2 Context

The main concept in the research work presented here is the virtualization of Residential Gateways (RGW). This equipment is currently installed on the customer premises which raises some problems to both, the service provider and the service subscriber.

The RGW is a relatively expensive and complex device that has to deal with all the complexity that services such as IPTV, Voice over IP (VoIP) and Internet, commonly named triple play, require. This complexity makes the RGW prone to hardware failures and/or misconfiguration situations that represent a considerable burden to service providers.

The specific location of an RGW, on the edge between the service provider access network and the subscriber home network, also represents a single point failure for service delivery.

The ongoing transformation on the way services and platforms are maintained and delivered to customers on the internet is also becoming more mature and stable. There has been a growing trend towards virtualization and we currently see more and more virtualized services and platforms being offered on the internet. Everyone who follows this market trend must have already heard some buzz terms like *Software as a Service (SaaS)* or *Platform as a Service (PaaS)*. These are the kind of cloud based services internet giants like Yahoo or Amazon currently offer.

Service providers are no different and the virtualization of key network elements such as the RGW is very likely to happen within the next times.

Consequently, our research tries to address all the previous issues with virtualization and proposes a full-fledged virtualization framework for residential gateways. The concept isn't novel[2] but this is to the best of our knowledge, the first end to end proposal on a complex environment such as a broadband access network with extensions to the service provider core network.

Further details will be covered on the following chapters since this chapter only pretends to give the reader a generic perspective on the topic.

1.3 Structure of this document

This thesis is split in more five chapters and one appendix, briefly described next:

- Chapter 2 - State of the Art

  This chapter presents the key concepts, technologies and related work that will support all the research on defining our framework for virtualized home gateways.

- Chapter 3 - Methodology and Proposed Approach

  This chapter presents the generic architectural approach that will support the virtualization framework implementation.

- Chapter 4 - Validation

  This chapter presents the prototypes created to validate the architectural proposal together with some experimental results.

- Chapter 5 - Evolution of the Research

  This chapter deals with the research planning and evolution over time, presenting a gantt chart with the most important activities.

- Chapter 6 - Conclusion and Future Work

  This chapter concludes the thesis and introduces the most important lines of work in future developments.

- Appendix A - Publications

  This appendix presents the project research in the form of research papers. The first paper addresses the specific work on virtualizing RGWs, while the second addresses a management framework for vRGWs.

- Appendix B - RGW Virtualization Testbed Setup

  This appendix is a detailed HowTo to setup the reference testbed used in our validation scenario.

2
State of the Art

This chapter addresses the key concepts, technologies and existing research used to define a full fledged end to end network virtualization architecture for Residential Gateways.

2.1 The Case for Virtualized RGWs

Cloud computing is seen by many seen as the next logical step in the evolution of computing models. The widespread availability of broadband network access is enabling the emergence of a multitude of new cloud sourced services, platforms and infrastructures.

The cloud term derives from the common used cloud symbol in network diagrams and represents nothing less than a set of technologies and concepts working together to allow the delivery and consumption of services hosted and supported by remote datacenters (providing dynamically scalable and often virtualized computing resources) to another service or an end-user, through a web browser, a lightweight desktop or mobile app. Cloud computing is also kind of a computation model similar to the old days [mainframe; dumb terminal] tuple but transported to a WAN level.

Looking into current access network deployments we see that FTTx network topologies are becoming predominant and in the next few years current DSL based topologies will be gradually upgraded to FTTx. With the deployment of FTTx, several new business opportunities arise and there's also lots of field addressable for improvement such as the triple play service platforms.

This research introduces a new network design for FTTx based broadband access network topologies. It demonstrates how the vRGW can be placed in the network, taking full advantage of today's cloud based service developments on providing *Everything-as-a-Service* and how several previously identified network impacts were overtaken or minimized by this approach.

As of today, the number of FTTx network access topologies is becoming the *de facto* standard in triple play service deployments. This opens a large opportunity window for operators to re-think how some of their service offers are deployed. Although the statement above is true, subscribers are not willing to pay more money for fiber than for copper access. Since fiber remains costly, operators seek other ways to cut on expenses, this could be achieved by changing the services or the physical Residential Gateway itself.

Looking into the physical RGW what we see is a device that is a trade-off between capabilities and cost which introduces several issues to service providers:

- Current RGWs are still expensive even considering the trade-off between capabilities and cost.
- Today's deployments put the RGW on the edge between the access network and the home network which creates an unavoidable single point of failure.
- The RGWs generate operational expenditure (OPEX), several call centers supported by service providers are RGW-related and it's often necessary to call for on-site support to verify the physical equipment and often replace it.
- Service provider time to market is often dependent on the unit manufacturer, to introduce new services to their subscribers which is obviously a serious penalty to pay.

- Management tasks are quite limited and if more than one manufacturer coexists that is even worse. Imagine what it is like to trigger a firmware update on millions of heterogeneous equipments just to be able to introduce a new service or feature.

To be able to reduce costs and mitigate the above issues operators could adopt a new disruptive approach for the Residential Gateway model. This model is the virtual Residential Gateway (vRGW). The vRGW would take advantage from the broadband network access high bandwidth to be pushed into the service provider network core fully replacing the physical unit as we know it today.

Basically the only thing that remains at customer premises is a simple unmanaged device, mainly for simple bridging purposes.

Figure 2.1 illustrates this approach.

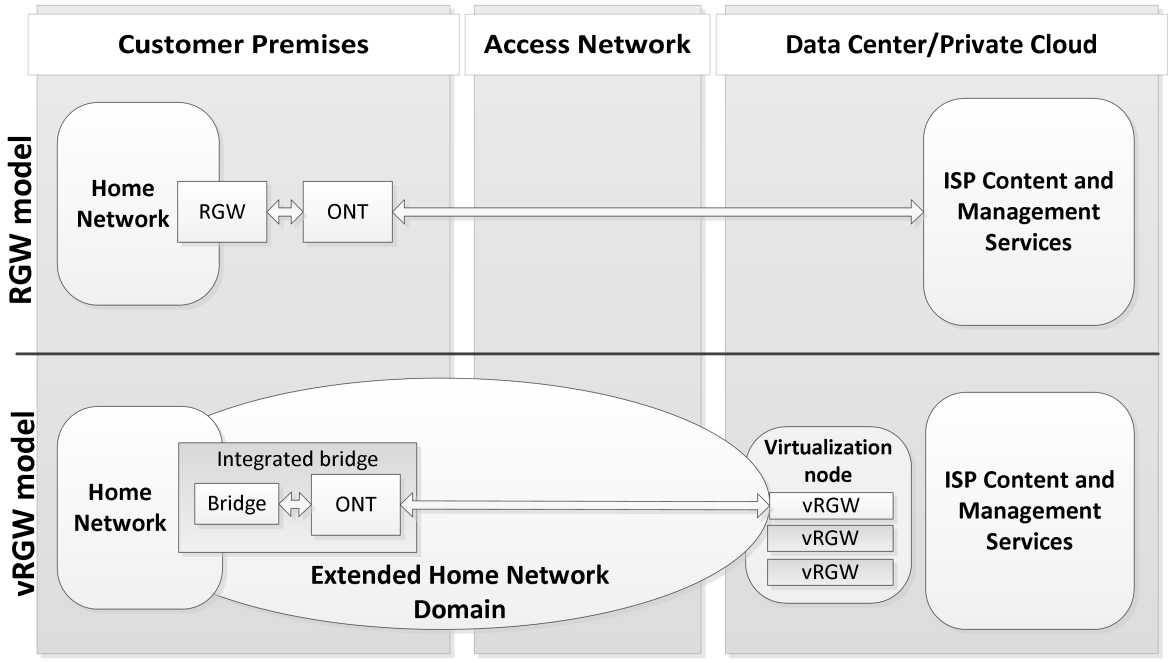


Figure 2.1: Classic RGW vs. Virtualized RGW.

Moving the RGW into the service provider network core will bring several benefits to both parties, to the service provider and to the subscriber:

- The service provider would be able to lower its capital expenditure (CAPEX) reducing operational expenditure (OPEX) at the same time since it would be no longer necessary to deploy a physical RGW. Savings on RGW-related call centers and with on site support are also envisioned. Outsourced maintenance would also be drastically reduced.

- To full support the vRGW, service providers would sill need to maintain a basic and inexpensive unmanaged bridge with an integrated IEEE802.11a/b/g/n access point along with the ONT unit at costumer premises due to the nature of some unmovable services/functions.

- Other functionality besides bridging could be thought as a virtual appliance deployed inside the service provider network. Such a solution could take advantage from the elastic features available in a data center infrastructure to cloud source the new vRGW which would also simplify and improve management operations dramatically.

- The pre-existent single point of failure would no longer exist or would be minimized in large extent, the service provider could now offer each subscriber a virtual instance of an RGW, while having backups on a pool of suspend, ready to kick in instances. This would make network downtimes residual since every defective running instance could be immediately replaced by the last good snapshot.

- The operator can now deploy new services in shorter windows in time, making its time to market more effective and efficient because the dependencies to specific manufacturers and unaligned time to market schedules between the operator and a specific equipment manufacturer would no longer exist.

- Today there is also a push for green telecoms and if service providers adopt the vRGW proposed architecture, it is possible to predict some energy savings and optimization by using the most efficient server farms available in current data center facilities. Thus it is also expected some improvement regarding the carbon footprint.

- In the end, the subscriber would benefit from the new shorter cycles regarding the introduction of new services and from a much more flexible and reliable service platform.

The research in this thesis, proposes a full-fledged framework for virtualized home gateways, addressing issues such as how to extend the home network, from a logical point-of-view, from the customer's premises into the service provider data center or the concrete virtualization of the RGW at the data center.

Another important issue is related to management of virtual residential gateways in a standalone scenario but probably more important on hybrid deployment scenarios where both physical and virtual RGWs exist.

The following section addresses the main topics on RGW management.

2.2 Management of Virtualized Gome Gateways

As already mentioned fiber-based access network deployments are becoming more predominant each day. This fact enables several improvements to the way how services are

currently delivered. This improvements are mostly related with management, maintenance and with the service nature itself resulting on more added value for both, the service provider and its subscriber.

The RGW stands right in the edge between the access network and the home network which implies on-site management more often than desired by service providers with all the costs that such interventions represent as discussed previously. Management of Residential Gateways (RGWs) have always been a complicated challenge for triple play service providers due to the sensitive positioning of this key network element on a triple play deployment.

The introduction of virtual Residential Gateways (vRGWs) has eliminated or largely mitigated almost every major management problems identified in traditional triple play deployments with physical devices. Therefore services such as DNS, DHCP, NAT, routing, firewalling, and added-value service management such as IPTV or VoIP can now be moved into the service provider private cloud in the form of a vRGW. This type of RGW services are the cause of most of the reported incidents in current triple play deployment scenarios, which often leads to on-site maintenance operations. Thus moving those into the service provider private cloud enables faster responses to reported incidents with an overall increase on efficiency. Nevertheless, we must not forget that it will always be necessary to bridge the local network devices (computers, set-top-boxes, telephones, etc.) with the access network and IEEE 802.11 wireless connectivity must also be assured locally. However that is not a big issue, because all the complex logic is now moved away from the customer premises.

The concept of a virtual gateway may have introduced significant advantages to triple play service providers by all the already mentioned reasons, however any service provider already selling triple play services won't want to throw away all the investment that has already been done, replacing all the deployed physical RGWs by virtual network elements. Furthermore, every service provider would have by now a more or less complex network management system, they want to preserve and maintain. The majority part of the current network management systems sit on top of CPE Wan Management Protocol (CWMP) for management purposes. Usual management tasks can range from a firmware update to group joins on a specific multicast stream to provide a new IPTV channel to the end user.

In this context it would be very interesting to provide integrated management of physical and virtual RGWs in a transparent way to operators. On our research we explore the idea of integrated management of traditional physical RGWs and the new virtual RGWs using CWMP as the management protocol. This approach should provide total transparency to the operator on the execution of regular management and operation activities with both network elements.

2.2.1 CPE WAN Management Protocol

CWMP has first emerged as a solution for management of DSL modems, making secure auto-configurations, diagnostic routines, software/firmware management and monitoring tasks possible on such devices. On a later stage CWMP was gradually extended to support residential gateways and also a large panoply of other customer

premises equipments.

CWMP is specified by TR-69 (where TR stands for Technical Report) and defined by the Broadband Forum. It is a protocol to be used between an Auto Configuration Server (ACS) and a Customer Premises Equipment (CPE). The ACS can be defined as a remote management server. TR-69 main objective is to provide remote management functions and to establish mechanisms to enable the auto configuration of a CPE in a secure way. Thus TR-69 specifies the generic requisites that its management functions should implement in order to be applied to any TR-69 compliant CPE.

CWMP is based on Simple Object Access Protocol (SOAP) web-services, it defines an API which stands on XML Remote Procedure Calls (XML-RPCs) and standardized data models for several types of CPEs. A secure layer can also be applied to the management operations through the usage of Secure Socket Layer (SSL) or Transport Layer Security (TLS).

2.2.2 Extensible CWMP

The eXtensible CWMP (X-CWMP[3]) is an extension to common CWMP and was initially developed to surpass the problem of being unable to remote manage CPEs not CWMP compliant inside home networks on triple play deployments.

Basically the interest for common CWMP as a management protocol is kept in service providers and mainly RGW manufacturers while most of the network equipment inside home networks speaks different protocols and doesn't implement a pure CWMP stack.

At this point X-CWMP was proposed as a mediation layer between standard CWMP on a RGW and other non compliant CWMP devices inside the subscribers home network enabling remote management on this non compliant devices and delivery new added-value services too.

The X-CWMP introduces two new entities with it, the master agent and subagents. X-CWMP has been defined to offer a robust and secure communication mean between this two entities. The protocol defines a set of XML encoded messages which could optionally be transmitted in a secure form by using SSL. Every action is confirmed with an acknowledgement (ACK) by both the master agent and the subagent which enables error checking and operation atomicity.

Since this concept deals with mediation between CWMP and X-XWMP messages, some message repetitions can be observed. Some examples are the, *SetParameterValues*, *GetParameterValues*, *Inform*, *AddObject*, *Download* among others. However, the X-CWMP protocol also defines a specific set of messages to guarantee the good functioning of the master agent and the subagents:

- ***RegisterSubagent*** This is the first message sent by the subagent. It specifies a register request with the authentication parameters and the data model to be used.

- ***UnregisterSubagent*** This message offers the subagent the possibility to send an unregister notification. From this point on, the ACS is unable to manipulate the data model of that subagent.

- ***InvalidRegistration*** Erro message indicating that the subagent was not registered on the master agent. This massage can appear with invalid authentication credentials or when there is a collision between data models of other subagents.
- ***ACK*** Notifies that the last message has been successfully received.
- ***RunMethod*** Enables a specific execution call on method from the subagent.
- ***Fault*** Indicates that other errors occurred.

Master Agent

The master agent is responsible for the decision on who to delegate a specific CWMP request. There is a *SubagentHandler* thread *per* each subagent connected to the CPE. After a standard CWMP request this thread processes it and translates it to X-CWMP. When finished the master agent tries to establish connection with the correspondent sub agent via socket, which enables X-CWMP message exchanges between both.

X-CWMP Subagent

The purpose of the X-CWMP subagent is to extend the functionality of the correspondent CPE. As such, it handles the incoming, parsing, processing and sending of X-CWMP messages to the master agent (more specifically, to his *SubagentHandler*). It also worths mentioning that each subagent is responsible for his data model manipulation.

2.3 Significance of vRGW Management

The introduction of this new concept of virtual Residential Gateways (RGWs) has brought a new challenge with it addressing manageability. Although virtual residential gateways could be managed by common cloud computing management frameworks like OpenStack[4] or oVirt[5], the specifics of a network element such as the RGW probably require a different approach.

On the other hand looking at current network management systems used by triple play service providers to manage physical RGWs, we see either proprietary or CWMP based frameworks. In order to keep current management systems, service providers would be at most, interested on extended functionally on their current systems, to start adopting virtual residential gateways.

Following this thought, it seems logical to seek for an integrated management framework that would manage both, physical and virtual RGWs in a transparent way.

CWMP is currently wide-spread by triple play service network infrastructures, as such, taking the key concepts out from the work on extensible CWMP, we could envision a CWMP based management solution for virtual Residential Gateways.

The work on extensible CWMP offers a key advantage that is the possibility of modelling simple CWMP subagents, clearly separating the CWMP specific routines

which are usually common to every CPE from the set of management functionalities associated with each managed service within the CPE.

In order to be able to manage virtual RGWs that do not come with an embedded CWMP stack, we could think on a CWMP subagent and on a master agent, both running on each virtualization node (hypervisor level) that would speak CWMP to the ACS and would execute system calls on the virtualized instances by means of an XML-RPC interface exposed by each virtual RGW for integration purposes.

An integrated management architecture for both physical and virtual RGW instances is presented Chapter 3, section 3.2.

2.4 Related Work

At an industry level there are various lines of thought on how the residential gateway (RGW) should be deployed and maintained for triple play service offers. The Home Gateway Initiative (HGI[6]) proposes two main approaches. Both continue to rely on the idea of providing each individual subscriber with a more powerful device.

- The support for a common application execution environment (e.g. the framework specified by the OSGi alliance), to enable easier implementation of new services in the Residential Gateway device[7].

- A standardized architecture, in order to lower costs through an off-the-shelf Residential Gateway.

This ideas from the HGI do not introduce anything relevant to the existing panorama. Basically these proposals only address the lack of resources identified when the service provider wants to deploy a new service to their subscribers and is faced with a problem: The already deployed RGWs do not have enough resources to support it.

The idea of having a common application execution environment is interesting but would not work on the long term and customers are also not willing to pay more, only to have a good enough device in the mid-term.

Above all this, service providers still have to account for the same or higher OPEX costs and would not be able to lower their CAPEXs.

Other studies[2] set out several different approaches to rich the same end, having a virtual Residential Gateway deployed inside the service provider network replacing the current physical device at the customer's premises.

One set of proposals introduces the idea of having an hardware based implementation of the vRGW. The virtualization could be accomplished at several different locations in the service provider broadband access network:

- **Integrated on the access node**

  This approach could be good because packet processing would be near the subscribers and distributed for several nodes but on the other hand it would force an hardware upgrade at the access node with unmeasurable costs for the operator.

  It would also be extremely difficult for network manufacturers to provide new Network Elements (NEs) with the required capabilities for such a virtualization scenario.

- **Integrated on the Broadband Network Gateway**

  This scenario has the advantage of keeping the network design unchanged although it is expectable that the broadband network gateway (BNG) capabilities would not be enough to account for the virtualization of several thousands of vRGW instances thus leading to another upgrade with the same consequences already identified in the previous item.

- **As a stand-alone network element**

  This option would not interfere with any of the already deployed NEs but it would add a new point of failure on the network and it would be another NE to be accounted by the Network Management System (NMS).

Regarding manageability for vRGWs, there is no previous identified work. Consequently a CWMP based integrated management framework, is to the best of our knowledge, the first one to be proposed that addressed at the same time, management of physical and virtual RGWs in a transparent way. However the general concepts on CWMP management for physical RGWs came from the work by Cruz, T. et al.[3] with their proposal for an X-CWMP agent extension framework for CWMP.

The same author also have a concrete proposal[8] for CWMP based management using decoupled CWMP agents, on Off-the-Shelf SIP Phones in Domestic and SOHO Environments which better demonstrates how to use the X-CWMP agent extension framework.

CWMP Management for home network devices is also covered by several technical reports from the broadband forum, namely TR-111[9]. TR-69[10] on their latest amendment 4 also addresses some of the concepts used on our CWMP based integrated management framework, namely annexes F and J.

The notion of device proxying based on CWMP-compliant gateways is also not novel. TR-69 Annex J covers CWMP Proxy Management and offers some interesting approaches regarding the CWMP proxy concept adopted by our implementation.

The broadband forum idealized the proxy concept for gateways (physical RGWs) but when vRGWs come into the equation this concept clearly makes sense when adapted for to virtualization nodes on a private cloud deployment scenario such as on our proposal.

TR-69 annex J still addresses the requirements for the data model and how to handle the proxied device availability.

3

Methodology and Proposed Approach

This chapter is a logical succession from the state of the art chapter and presents the methodology and proposed approach used to define the aforementioned full fledged end to end network virtualization architecture for Residential Gateways.

3.1 Proposed Network Architecture

Linking the home network from the customer premises to the private cloud on the service provider data center may seem an interesting proposal, but is a major challenge primarily based on the problem of scalability and management imposed by the hundreds of thousands of vRGWs involved.

Nevertheless, two reference frameworks from the Broadband Forum[11] would be a good reference starting point to go through with this objective. Both frameworks were proposed for a different target but most of the proposals are an excellent ground start to support our virtualization intentions.

The Broadband Forum has defined Ethernet-Based Broadband Aggregation scenarios over DSL access networks (TR-101[12]) and Gigabit Passive Optical Network (GPON) access networks (TR-156[13]).

TR-101 provides an Ethernet-based architecture that has become a global standard for triple play deployments for residential and business customers using DSL as access technology. Although, many of the specifications in TR-101 are not exclusive of DSL network elements.

The massification of broadband network accesses mainly supported by GPON urged an update to TR-101 in order to standard the same specifications in GPON scenarios. The result was TR-156.

3.1.1 Broadband Forum VLAN Aggregation Topologies

Essentially looking into GPON deployments, the Optical Line Termination (OLT) and Optical Network Terminal (ONT) share the responsibility of handling VLAN requirements in the access node as specified by the Broadband Forum. TR-101 identifies three VLAN topologies:

1. **Service VLAN (N:1).** This topology defines a single VLAN per service, which carries traffic to all subscribers. Each new service requires a dedicated VLAN (S-VLAN). Current triple play service provision is mostly sustained by this specification having its multicast IPTV delivered in the same VLAN to all subscribers. For GPON environments, TR-156 adds some extra complexity into the specification. The ONT adds an S-VID (S-VLAN ID) or translates an incoming S-VID tag for upstream traffic, so that there is always an S-VID between the aggregation node and the access nodes. The aggregation node will allow any upstream frames with an S-VID to pass-through. The downstream is essentially the opposite operation, with the aggregation node passing through the S-VID. The access nodes will remove or translate the tag and then forward frames to its associated U interface (ONT) into customer premises. Figure 3.1 depicts the specifications for both technologies, DSL and GPON.

2. **Customer VLAN (1:1).** In this specification, there is one VLAN *per* subscriber, which carries all its traffic. This approach is quite interesting for what we want to do in our vRGW based service delivery mainly because it imposes a

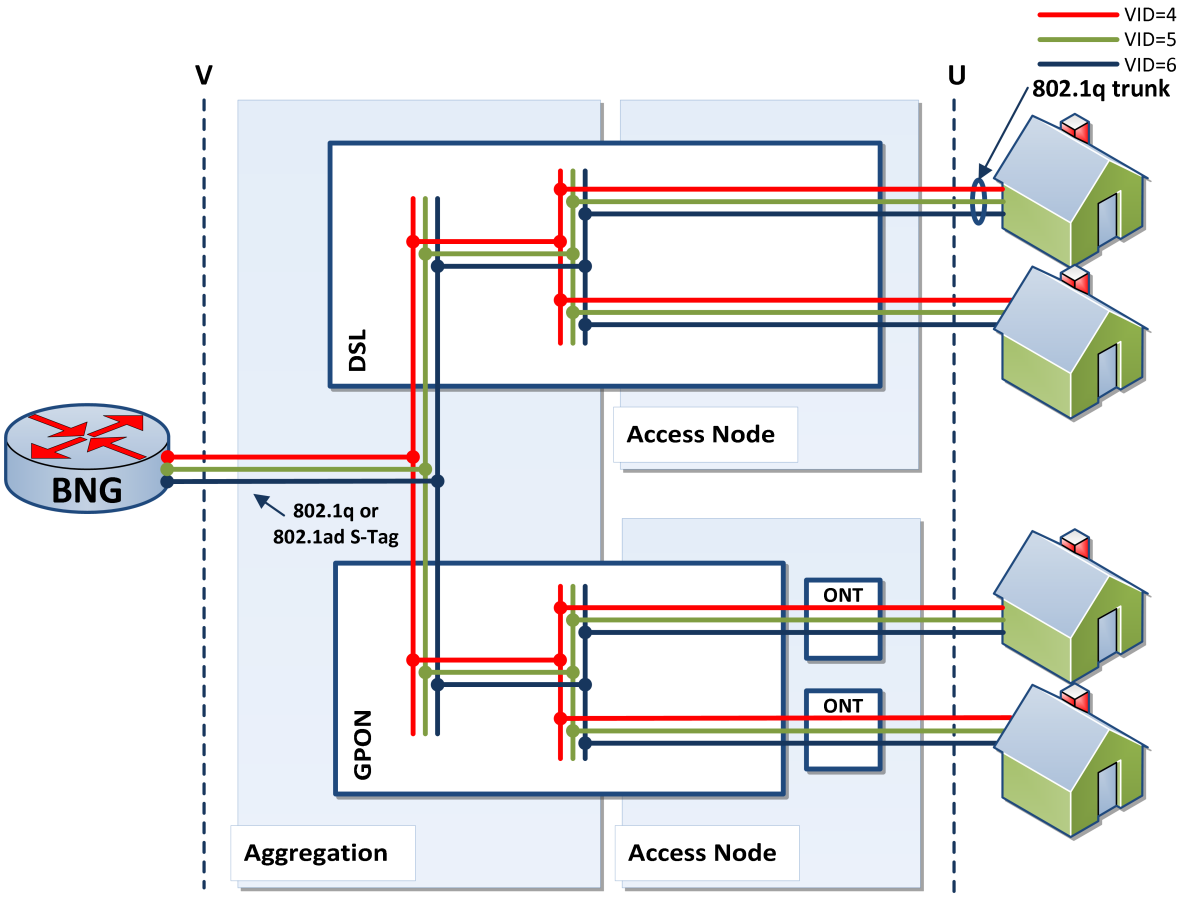


Figure 3.1: VLAN per service model (adapted from [1]).

1:1 mapping between services and Customer-VLANs (C-VLANs), trusting on the edge router to manage thousands of VLANs. In spite of that this approach also has a big drawback in it, IPTV service delivery would not be efficient because it would force the delivery of multiple/equal IPTV streams to all subscribers.

Looking into GPON specifically, the ONT maps each 1:1 VLAN into a unique U interface and each U interface maps into one or more 1:1 VLANs. The ONT always adds a tag to untagged frames or translates an incoming Q-Tag in the upstream direction. Tag assignment at the V interface can be done in two different ways.

(a) The *single tagging*. The ONT acts on single-tagged VLANs at V and adds an S-VID or translates an incoming tag into an S-VID while the aggregation node passes through the tag. It still worth mentioning that the 12-bit VLAN identifier only supports up to 4095 subscribers.

(b) *Double tagging.* VLAN stacking[14] (initially standardized by 802.1ad[15] and later incorporated into the 802.1Q standard) makes it possible to increase the number of VLANs beyond the theoretical number of 4095.

In this case the ONT acts on double-tagged VLANs at V and assigns a C-VID (C-VLAN ID) or translates an incoming tag into a C-VID while the aggregation node adds the S-VID. Service providers can recur to this specification to enable subscribers service delivery using multiple VLANs *per* subscriber. Basically the stacked VLAN (802.1Q-in-Q) frames from each subscriber will travel across the service provider network core without major service impacts or performance penalties also keeping the traffic from each subscriber completely isolated from each other.

Figure 3.2 better demonstrates the situation.

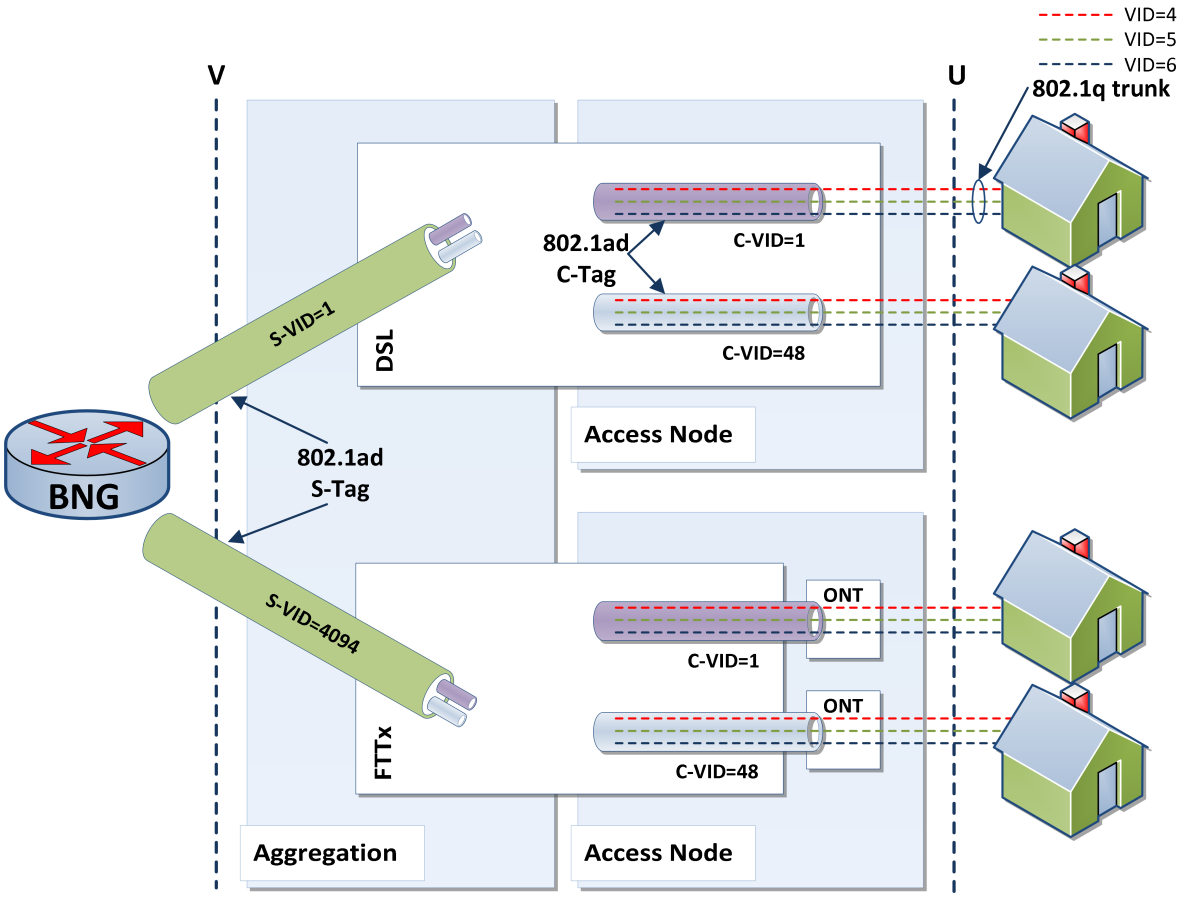


FIGURE 3.2: VLAN per subscriber model (adapted from [1]).

The downstream is essentially the opposite operation, with the aggregation node removing an outer tag (if there is one present). The OLT will removes or translates the tag and then forward frames to its associated U interface.

3. **Customer VLAN with Multicast VLAN.** This is the best option to feet our virtualization architecture intents. This approach basically picks what's best in the two previous specifications. While keeping a 1:1 mapping the previous drawback of having multiple IPTV streams is solved introducing a shared multicast VLAN (M-VLAN[16]).

3.1.2 Proposed Architecture for Support of vRGW

To enable support of vRGWs, our network architecture proposes a "distributed centralization" of this network elements using one or more data center facilities located in the service provider network to implement a private cloud network model. The network architecture supporting the introduction of vRGWs is presented in Figure 3.3. The scenario depicted by Figure 3.3, proposes the hosting of vRGW instances across two different data centers (DC#1 and DC#2) to better represent scalability possibilities. Thus the aforementioned designation of "distributed centralization".

Each virtualization node is responsible for a set of vRGW instances. Each virtualization node is connected to the network by a mini VLAN trunk representing the I/O network traffic of every individual vRGW instance running in that specific virtualization node (i.e., the corresponding set of C-VLANs).

Each mini VLAN trunk is aggregated into a larger sub-trunk representing all the C-VLANs on a specific data center. This sub-trunk is then encapsulated into MPLS Pseudowires (PW) at one of the MPLS edge routers in the service provider network core. Those PW are then transported to other edge routers, also known as Broadband Network Gateways (BNGs).

At the exiting BNGs, each PW is converted again into VLAN sub-trunks, each of them connected to the corresponding network access node.

At each individual access node, each VLAN sub-trunk is divided into smaller C-VLAN trunks carrying the traffic of each individual subscriber (separated by one VLAN for each service being delivered).

At the customer premises, on the ONT, each VLAN is untagged and mapped into a port in one unmanaged bridge with a wireless access point (the ONT and this bridge are not represented in Figure 3.3, for sake of simplicity). At this point all the equipment inside the subscriber LAN can connect to the stripped-down bridge and obtain a valid IP address from the DHCP daemon running on the vRGW. From this point forward, the subscriber network usage experience is similar to what exists today with a physical RGW.

This network architecture allows extending the logical reach of the subscriber LAN to the vRGW hosted at the service provider private cloud without major impacts on current service offers based in physical devices.

Moreover, the joint use of VLAN stacking, VLAN trunks and MPLS Pseudowires make it possible to handle, aggregate and encapsulate in a scalable manner the high number of VLANs required by thousands or millions of subscribers.

Considering the aforementioned Broadband Forum specifications, the proposed architecture would fit both, the Customer VLAN (1:1) and the Customer VLAN with

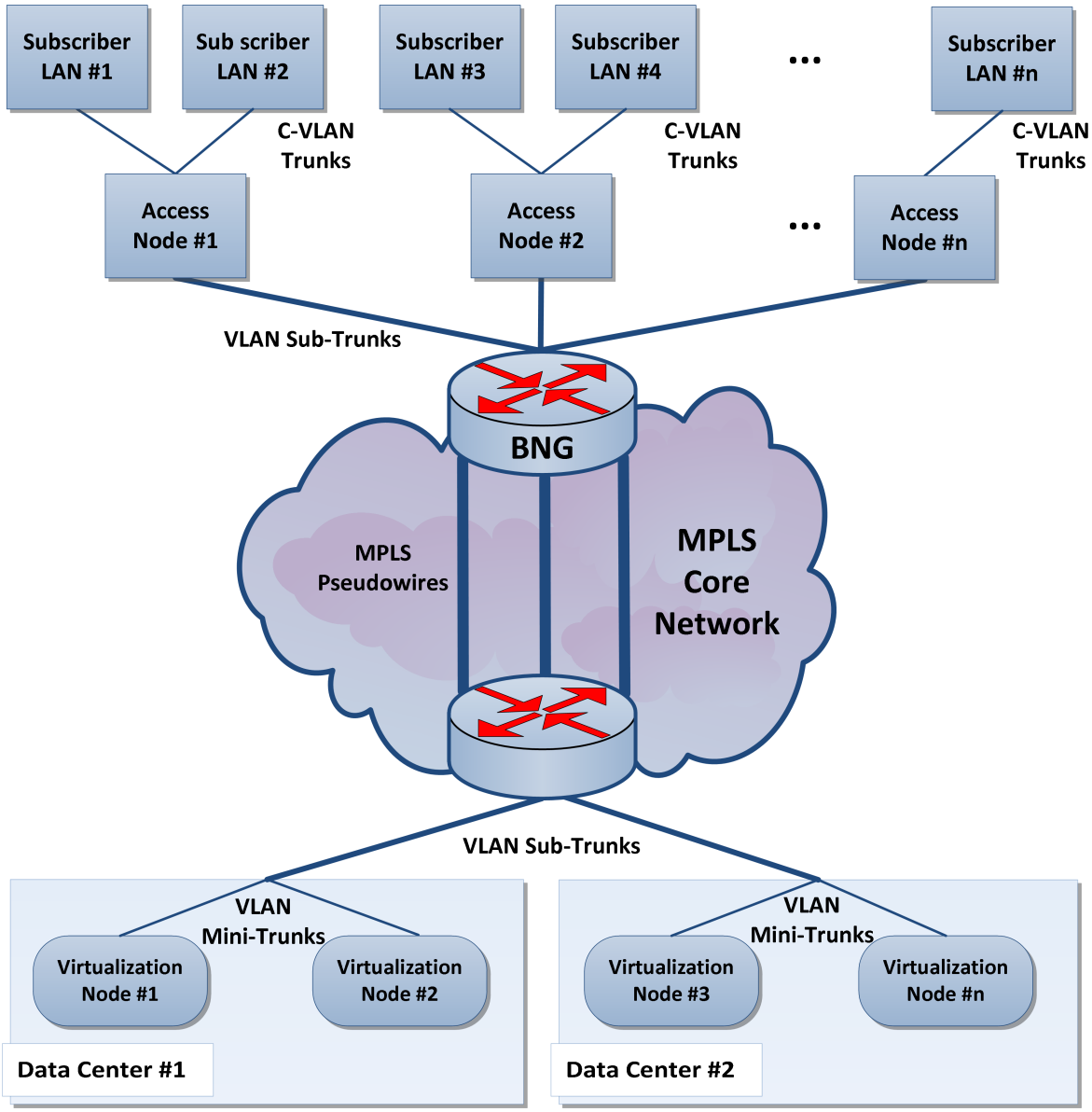


FIGURE 3.3: vRGW-enabling network architecture.

Multicast VLAN, the last would be the most efficient option as mentioned before.

Vertical Segmentation vs. Hybrid Approaches

With regard to the context of this research, the virtualization of the RGW in the service provider private cloud is achieved through the direct instantiation of virtual machines with the operating system being identical to that which exists today in physical equipments. The concept of service sharing between vRGWs does not exist at this point with the exception of our proposed management framework.

However, service sharing among vRGWs is something that could be greatly explored in future advances. It makes all the sense to us putting some services such the application level gateways (ALG) and management frameworks on a common layer shared between vRGWs. This concept could be seen as a vertical *vs.* hybrid approach as seen in Figure 3.4.

As already mentioned the hybrid approach concept is in fact already used by our research on the proposed management framework, to be presented further ahead.

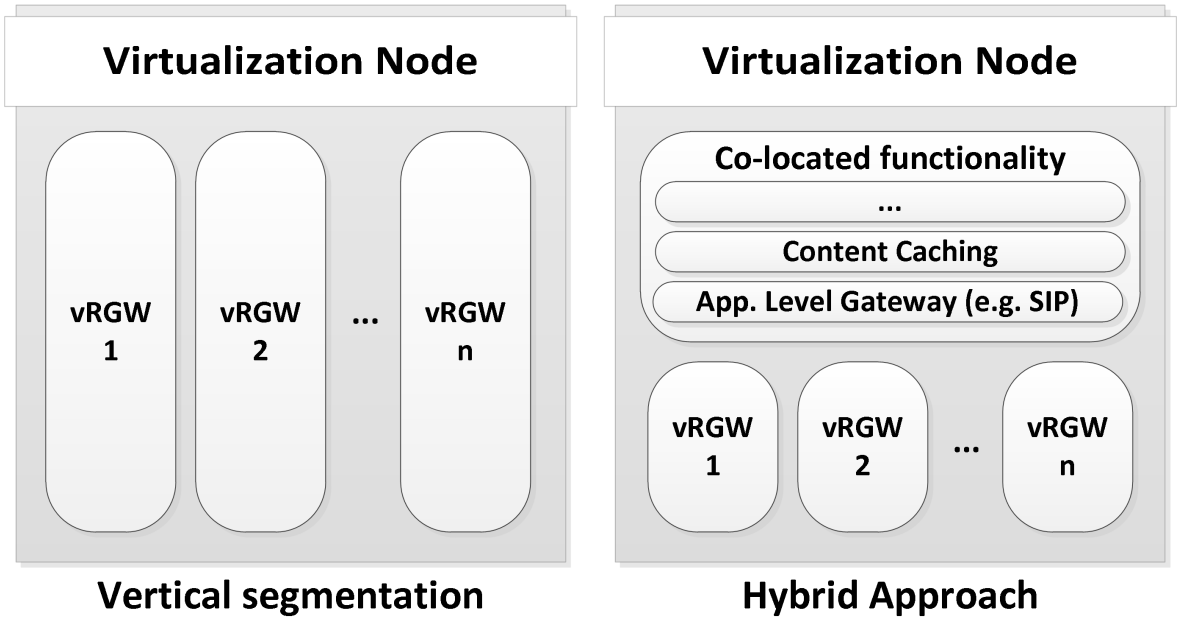


Figure 3.4: Vertical Segmentation vs. Hybrid Segmentation.

3.2 Proposed Integrated Management Framework

In order to take full advantage from the proposed network architecture, manageability is something that cannot be left out of the equation, being a fundamental part on operation support systems in current triple play service offers.

As mentioned before, management in most of the current triple play service offers is guaranteed by the CPE Wan Management Protocol (CWMP), which led us to adopt it in the definition of a new management architecture for virtual RGWs.

The proposed management architecture for vRGWs is in fact an extension to the existing CWMP management frameworks and will allow management of physical and virtual RGWs in a transparent way. The proposed architecture is depicted by Figure 3.5 that shows the usage of what can be called a CWMP proxy. This key component runs at each vRGW instance providing an XML-RPC interface to be used in typical CWMP operations.

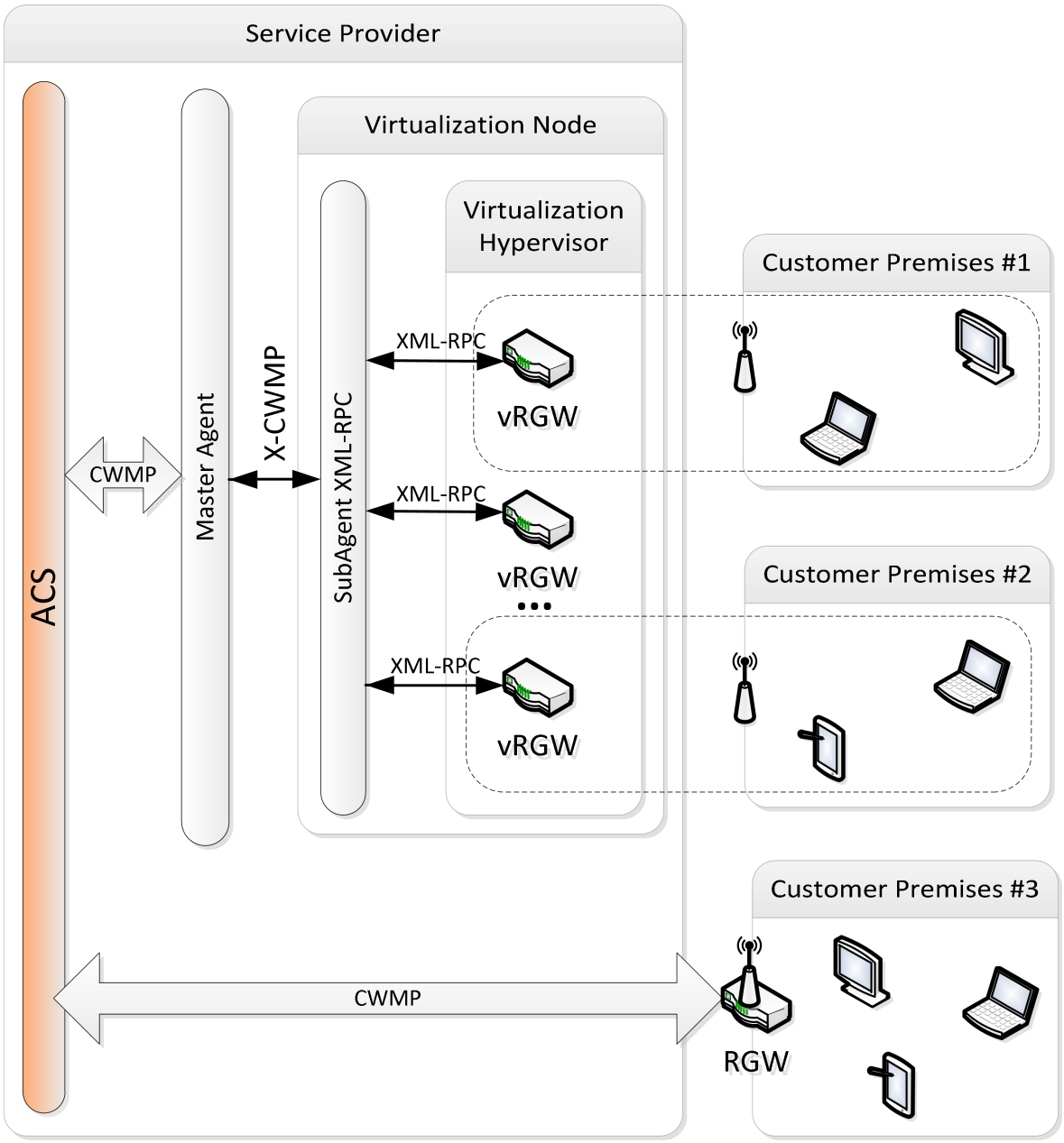


FIGURE 3.5: CWMP based Integrated Management Architecture.

The interesting part with this approach is that it takes advantage of that hybrid approach concept discussed earlier to provide full CWMP management. Thus having a complete native CWMP stack *per* each deployed vRGW instance becomes unnecessary. It also worth mentioning that a full native CWMP stack running on an embedded system can be extremely resource demanding, which is a common identified disadvantage in CWMP applied to physical devices (CPE).

The proposed architecture keeps all the CWMP logic inside the service provider

network infrastructure by means of CWMP agents. Taking advantage from the extensible CWMP framework discussed in chapter 2, section 2.2, it is possible to have one or several master agents running near the virtualization nodes.

In spite of everything the proxy implementation does not know anything about the CWMP protocol, it just exposes an interface to be used by CWMP agents in CWMP operations. As already mentioned, the CWMP logic is kept in the CWMP agents as seen in Figure 3.5. The master agent speaks CWMP with the auto configuration server (ACS) and X-CWMP with the XML-RPC subagent, which in turn speaks XML-RPC protocol with the proxies running inside each vRGW.

The proxy concept[10] makes the overall solution extremely flexible to be adopted by other devices, keeping the CWMP complex logic all in one place.

It worth mentioning that developing for embedded systems could be quite painful to normal programmers due to resource limitations so common on this type of systems. Moreover, cross compiling existent code proves to be a daunting task, sometimes more painful than developing everything from the scratch.

Finally a closer look on Figure 3.5 shows that traditional management of physical devices (RGW) still keeps transparent to the operator and the vRGW addition is seen as if it was another physical device regarding manageability tasks. The only observable difference is that the home network from the subscribers with a vRGW is logically extended into the service provider private cloud while the other subscribers continue to keep the physical RGW on the edge between their home networks and the service provider broadband access network separating the two networks. However this difference is completely invisible to management operations on both types of equipment.

4
Validation

This chapter takes the methodology and proposed approach described before and implements the prototypes to assess the overall viability in the presented frameworks. Each section ends with a discussion on the experimental results.

4.1 An Architecture for Virtualized Residential Gateways

4.1.1 Reference Testbed

In order to validate the proposed vRGW approach, we have built the proof-of-concept testbed illustrated by Figure 4.1, which emulates the data center environment and the Home Networks.

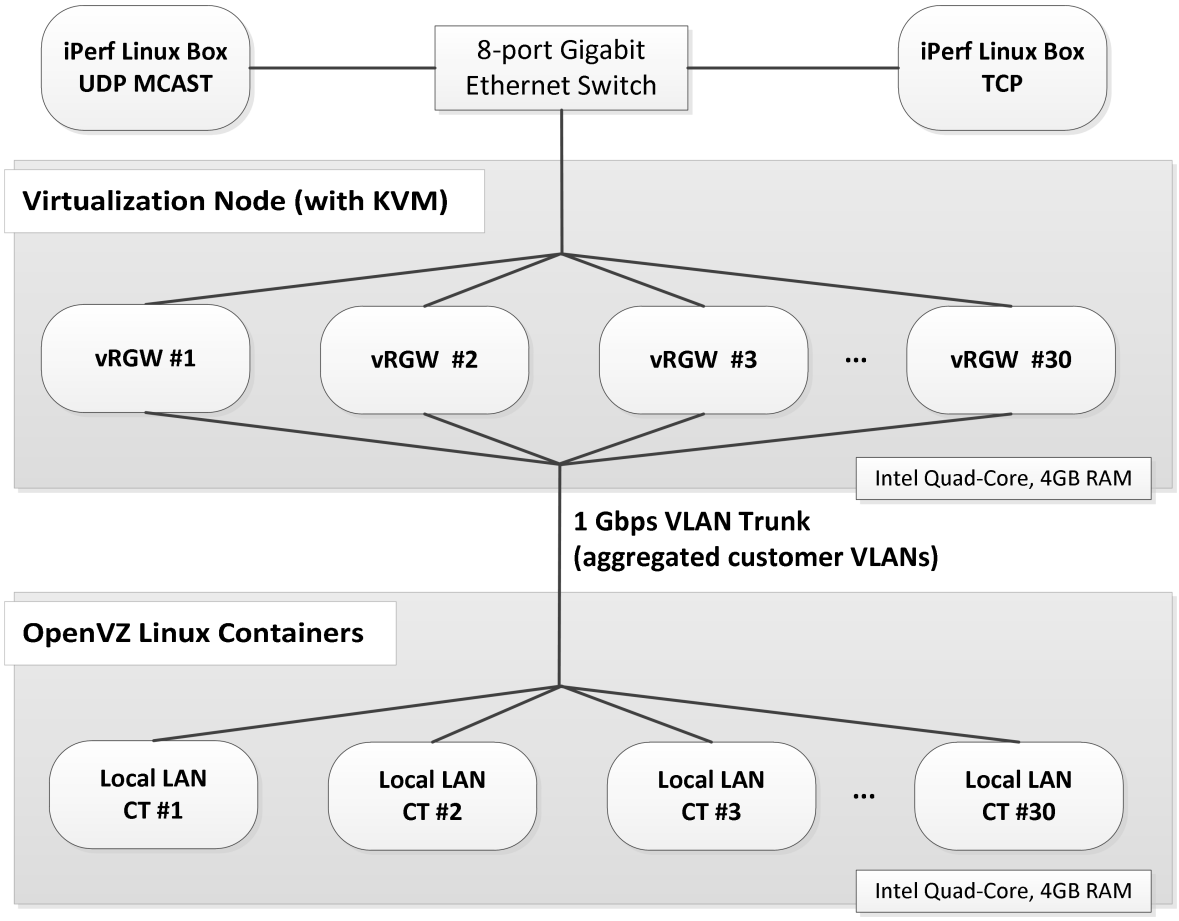


FIGURE 4.1: RGW virtualization testbed.

The virtualization node was based on an Intel Quad-Core server with 4GB of RAM, running CentOS Linux 6 x86_64[17] with KVM (Kernel-based Virtual Machine[18]) as the virtualization hypervisor and *libvirt*[19] as the management framework.

The interconnection between the Virtualization Node and the Home Networks was based on a 1Gbps Full Duplex Ethernet link, supporting one point-to-point VLAN trunk between the virtualization node and another CentOS 6 Linux x86_64 server running up to 30 OpenVZ Linux Containers[20] (LXC) to emulate up to 30 independent

subscribers (one per vRGW instance on the virtualization node). This VLAN Trunk aggregates the multiple C-VLANs involved.

All the details on how to setup this reference testbed can be found in appendix B.

The virtualization node was connected to a 8-Port Gigabit switch where two other Linux machines were running *iperf*[21], one of them configured as a UDP and UDP Multicast server source (to emulate VoIP and IPTV traffic, respectively) and the other as TCP generator (to emulate Internet connectivity and other general TCP traffic). The Quality of Experience Requirements for IPTV Services proposed by ITU for "Average Users" and "High Users"[22] (see Table 4.1) were used to model iperf-generated UDP traffic (IPTV and VoIP services). The "TCP Internet" traffic generator was not capped, using the remaining available bandwidth.

Table 4.1: Downstream bitrates for all-digital triple play.

Service	**Average User (Mbps)**		**High User (Mbps)**	
Internet		5.0		10.0
Telephony		0.1		0.1
SDTV (MPEG-4)	2 channels	3.0	2 channels	3.0
HDTV (MPEG-4)	2 channels	8.0	2 channels	16.0
Total		**16.1**		**29.1**

4.1.2 Virtualization of the RGW

The presented proof-of-concept prototype uses OpenWrt Backfire x86 RGWs[23], virtualized using the KVM hypervisor and the libvirt management framework. Experimental measurements were collected using *virt-top*[24] for showing stats of virtualized domains.

OpenWrt was chosen for practical reasons, since it is one of the two most popular opensource RGW implementations. OpenWrt is a mature and well-organized project, easy to adapt for out purposes (selection and management of components, support for x86 architectures, support for customization and can even be specially compiled to take advantage from all the potentialities offered by the KVM virtualization hypervisor). Besides, OpenWrt is in fact representative of a classical RGW (several commercial devices are actually based on OpenWrt).

KVM was selected because it is an opensource virtualization framework mature enough to be considered on pair with commercial platforms and not subject to legal benchmarking/disclosure constraints. It is compatible with management frameworks (such as oVirt[5], Open Stack[4] or Open Nebula[25]) and can support large scale virtualization architectures, including management of live migration, load balancing, provisioning, virtualization node life cycle, high availability, power saving and storage. Those features are not pertinent in our small proof-of-concept prototype but might become relevant in future developments.

4.1.3 Experimental Results

In order to assess the capability of our testbed, we conducted measurements varying the number of vRGWs and considering ITU-defined requirements:

- **For "Average Users":** 2 SDTV channels, 1 HDTV channel and 3 phone calls using the G.711a codec (which exceeds the 0.1 Mb/s specified by ITU for telephony).
- **For "High Users":** 2 SDTV channels, 2 HDTV channels and 3 phone calls have been considered.

Figure 4.2 presents the average throughput measured for each Home Network, with 1, 10, 15, 20, 25 and 30 vRGWs hosted by the Virtualization Node. Results show that the ITU reference requirements were easily surpassed, even with 30 vRGWs and the modest hardware used for the Virtualization Node.

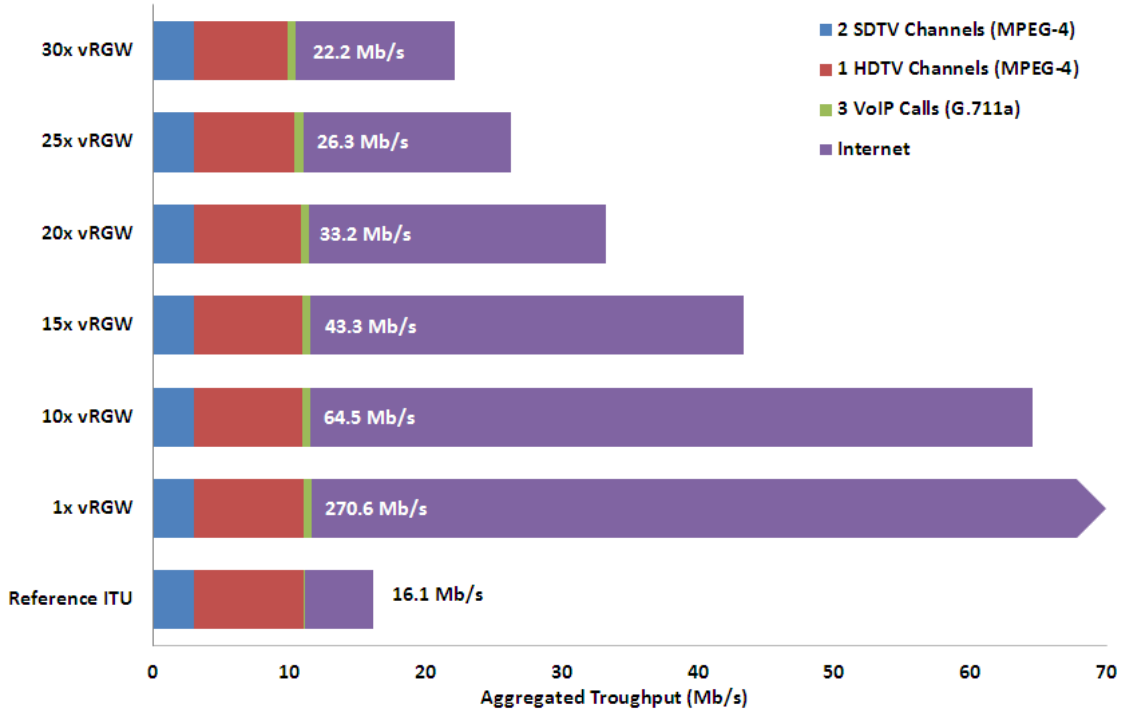


FIGURE 4.2: Measured Throughput per vRGW ("Average User").

The measured average throughput per Home Network achieved for the "High User" profile is presented in Figure 4.3. According to these results, our prototype should be able to support between 20 and 25 "High Users".

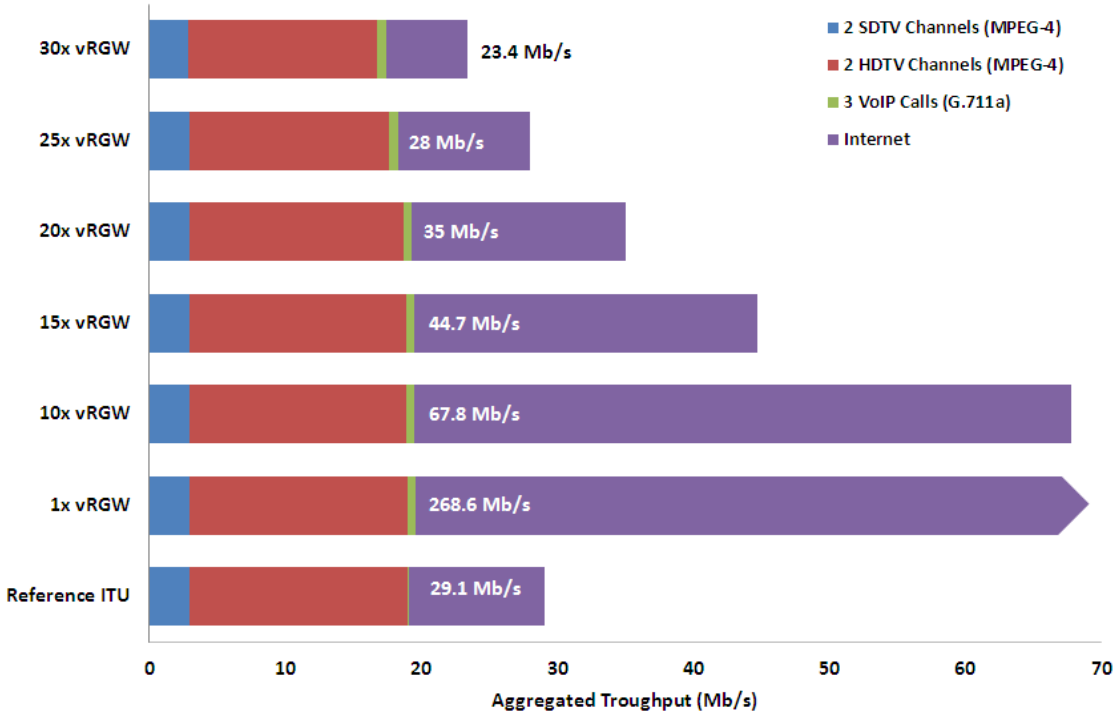


Figure 4.3: Measured Throughput per vRGW ("High User").

We noticed small but noticeable packet loss for the UDP traffic in the most demanding scenarios (many vRGWs). The three worst cases were: 4.30% packet loss (30 vRGW and "High User" profile), 2.35% (25 vRGW, "High User") and 1.57% packet loss (30 vRGW, "Average User"). This can be justified by the fact that we used no traffic prioritization, an unlikely scenario in production environments. Furthermore, such packet loss ratios can be handled by the forward error correction (FEC) mechanisms employed by IPTV platforms without any noticeable degradation of quality[26][27].

During the experimental tests we also monitored the Virtualization Node, in terms of CPU load and memory usage. Figure 4.4 shows the total CPU load used by the hypervisor of the Virtualization Node, for the "High User" scenario (represented as a percentage of the maximum CPU capacity). Apart from the single vRGW case, the average CPU usage for 10, 15 and 30 vRGWs under load fits between 75% and 85%. This headroom threshold could be explained by the fact that, besides running the virtualization hypervisor, the Virtualization Node also needs to handle the management of multiple VLANs with high load, running network I/O traffic on its VLAN trunk.

Figure 4.5 presents the average CPU load occupied by each individual vRGW (for the same "High User" scenario), represented as a percentage of the maximum CPU capacity of the Virtualization Node. Results are consistent with the hypervisor measurements of Figure 4.4, showing no signs of remarkable overhead at the level of the

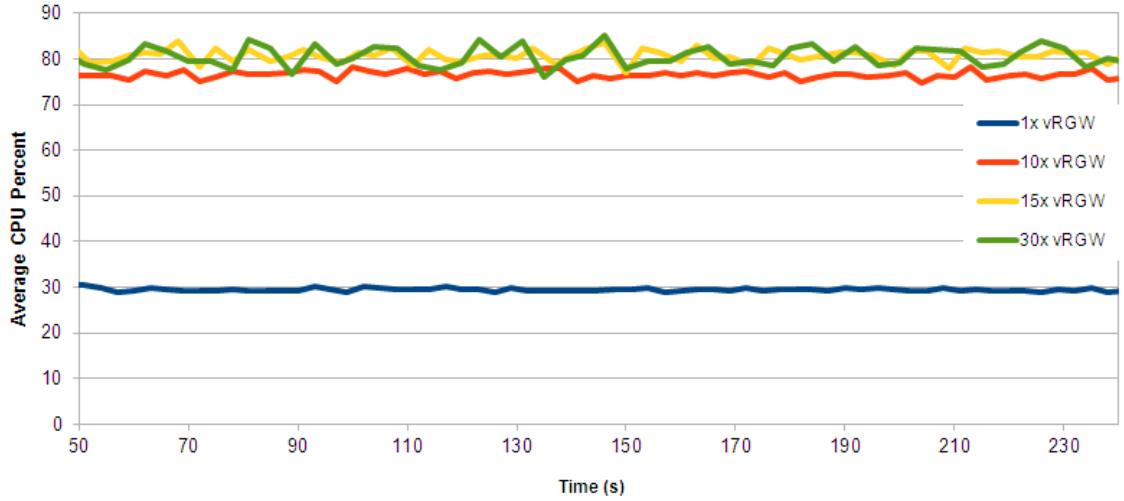


FIGURE 4.4: CPU used by the Hypervisor ("High User").

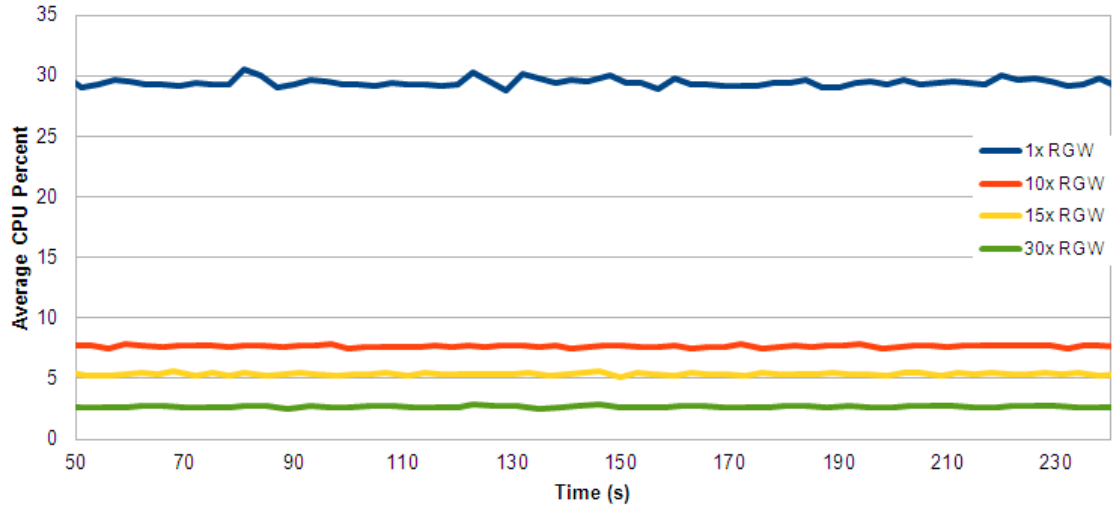


FIGURE 4.5: Average CPU used per vRGW ("High User").

hypervisor.

In terms of memory allocation, each vRGW OpenWrt instance was configured with 32MB of RAM, totaling up to 960MB (30x32MB) of allocated memory on the hypervisor. During the tests each vRGW consistently used around 18% of available memory (more or less 6MB), showing that RAM was not a key bottleneck in the specific system we used.

Finally, it should be stressed that the vRGW capacity per Virtualization Node, as hinted by presented measurements, is strongly capped by the modest hardware we used (old-generation Intel Q8400 2.66GHz, 4 GB of RAM, 2 non-optimized Gb/s network interfaces). Commercial deployments are likely to achieve dramatic capacity improvements simply by careful specifying and tweaking critical hardware components, as well as by using the resource consolidation tools provided by data-center grade virtualization platforms.

4.2 A Management Framework for Virtualized Home Gateways

4.2.1 Experimental Testbed

In order to validate our integrated CWMP management framework, the testbed shown in Figure 4.6 has been built to emulate the vRGW management operations only, although CWMP management of physical devices could be easily achieved by just connecting the ACS to such an RGW device.

The virtualization node was based on an Intel Quad-Core server with 4GB of RAM, running CentOS Linux 6 x86_64 with KVM (Kernel-based Virtual Machine) as the virtualization hypervisor.

The ACS ran on another CentOS 6 Linux 6 x86_64 server with the same hardware characteristics. The ACS consisted on a JAVA application executing as a standalone application on a simple java virtual machine (JVM), although it could be made available on a java application server (AS) such as JBoss[28] or Glassfish[29].

At the virtualization node we had two more standalone java applications implementing extensible CWMP, one for the master agent and the other was the XML-RPC[30] subagent which implemented a simple CWMP data model for management of OpenWrt vRGW instances.

At the KVM hypervisor[31] we have setup a group of thirty OpenWrt vRGW instances. One CPU core and 32MB of RAM have been given to each running instance.

At each OpenWrt vRGW we also had a python daemon listening to system call requests coming from the extensible CWMP agents on the virtualization node. As discussed earlier, the communication protocol used between the XML-RPC subagent and the python "CWMP proxy" was plain XML-RPC.

4.2.2 CWMP Proxy Concept

The presented proof-of-concept prototype uses OpenWrt Backfire x86 RGWs, virtualized using the KVM hypervisor and the libvirt management framework.

Experimental measurements were collected using *top*[32] for registering the python CWMP proxy daemon CPU and memory percentage utilization footprint at each vRGW instance and *tcpdump*[33] on the virtualization node to capture traffic between

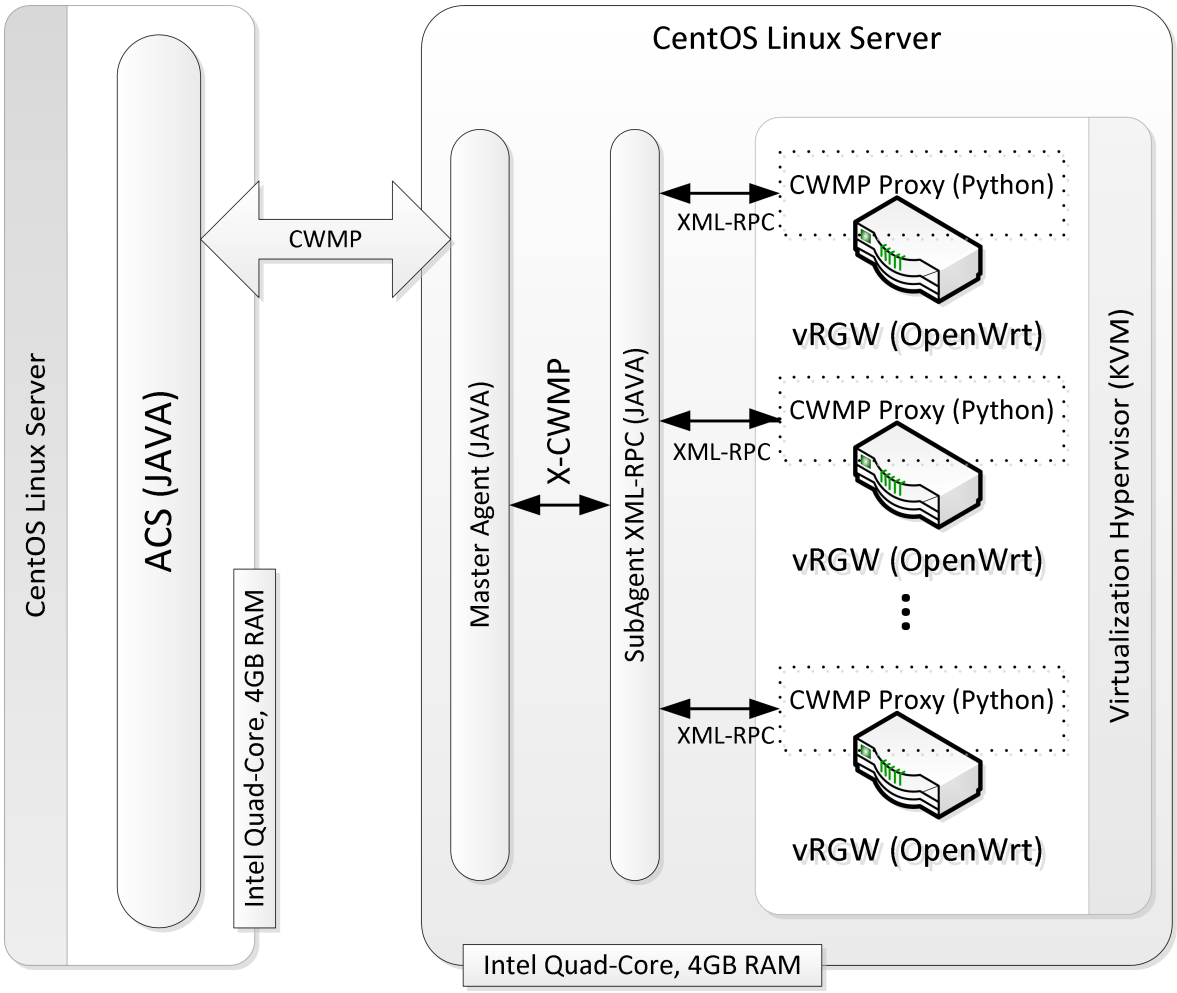


FIGURE 4.6: CWMP based vRGW management testbed.

the ACS server, the virtualization node and each running vRGW according with the scenario from Figure 4.6.

Moreover, in order to register typical CWMP operation durations, both the ACS server and the extensible CWMP agents all JAVA applications, were customized with a nanosecond precision timer for operation duration measurements.

OpenWrt and the KVM virtulization hypervisor have been chosen for practical reasons, most of them already discussed on the other prototype here.

Coming back to the CWMP proxy concept, it worth mentioning that on a first approach we could think on developing a native CWMP stack for OpenWrt, although it might seem a logical step, porting code to embedded systems such as OpenWrt is not an easy task and requires lots of effort. Besides that, the need for a CWMP management stack in OpenWrt OS was a great opportunity to go for the hybrid approach *vs.* vertical segmentation design discussed earlier.

Furthermore, there was a java based CWMP framework already available and since there was no support for a java virtual machine (JVM) in OpenWrt, we did not see

any logic in porting the full stack to OpenWrt. Instead, the logical approach would be taking advantage from the extensible CWMP framework also available and develop a new subagent responsible for handling the CWMP communication between the vRGWs and the ACS as seen in Figure 4.6.

Looking at the implementation requirements in OpenWrt *vs.* the available software framework options, we came across with an idea of using a well known and scalable scripting language (Python) to develop our "CWMP proxy" which would basically consist on a OpenWrt daemon that would run at system startup, listening to XML-RPC requests coming from our extensible CWMP subagent.

All the CWMP logic would be implemented by our extensible CWMP agents while the "CWMP proxy" in OpenWrt would be just waiting for system call requests.

The proposed "CWMP proxy" for CWMP management of vRGWs also emerges from some noted facts:

- Both OpenWrt and libvirt already offer good management solutions.
- OpenWrt offers a great package management solution similar to what's currently found in desktop Linux distributions.
- Libvirt provides a great low level management interface to the KVM virtualization hypervisor.

While analysing the available software frameworks in OpenWrt we have elected Python[34] to implement our "CWMP proxy". Pyhton seemed well-supported and had some interesting libraries already ported to OpenWrt, namely the XML-RPC library.

With the final "CWMP Proxy" implementation completed, we have come to one conclusion:

- Our pythton implementation proved to be a wise choice as the results presented here can tell.

4.2.3 Experimental Results

In order to assess the capability of our testbed, we conducted measurements varying the number of vRGWs from only one to thirty while registering operation response times to the request for changing the LAN interface address:

```
ifconfig br-lan 192.168.255.1
```

Table 4.2 that follows bellow shows the descriptive statistics for the described bulk operation involving thirty vRGWs and twelve experiences.

The chart from Figure 4.7 shows execution response times for a single request on one vRGW alone *vs.* the average response time *per* vRGW while performing a bulk operation with thirty vRGWs involved. The error bars on the same chart represent the 99% level of confidence for the mean on the 30x vRGW bulk request operation, while standard deviation is used for the single request.

The reason not to use standard deviation in both cases is because standard deviation is a dispersion measurement with little resistance, mainly because it is very sensitive to

the influence of too large or too small values and as seen in Table 4.2 the maximum and minimum values in the sample were 124.832 (ms) and 18.314 (ms) respectively, which results in a standard deviation value of 14.922(ms). However calculating the 99% level of confidence for the mean we have gotten only 2.037 (ms). Thus in this particular case the 99% level of confidence for the mean was a better representation of reality.

Moreover, it also doesn't make sense to use the level of confidence for the mean in both cases, because the sample for the 1x vRGW single request was too small and bellow thirty, which is a common ground reference value for the calculation of the level of confidence for the mean.

TABLE 4.2: Execution Response Times Descriptive Statistics.

	Time (ms)
Mean	38.386
Standard Error	0.786
Median	34.210
Mode	27.151
Standard Deviation	14.922
Sample Variance	222.664
Interval	106.518
Minimum	18.314
Maximum	124.832
Count	360.000
Confidence Level (95%)	1.547
Confidence Level (99%)	2.037

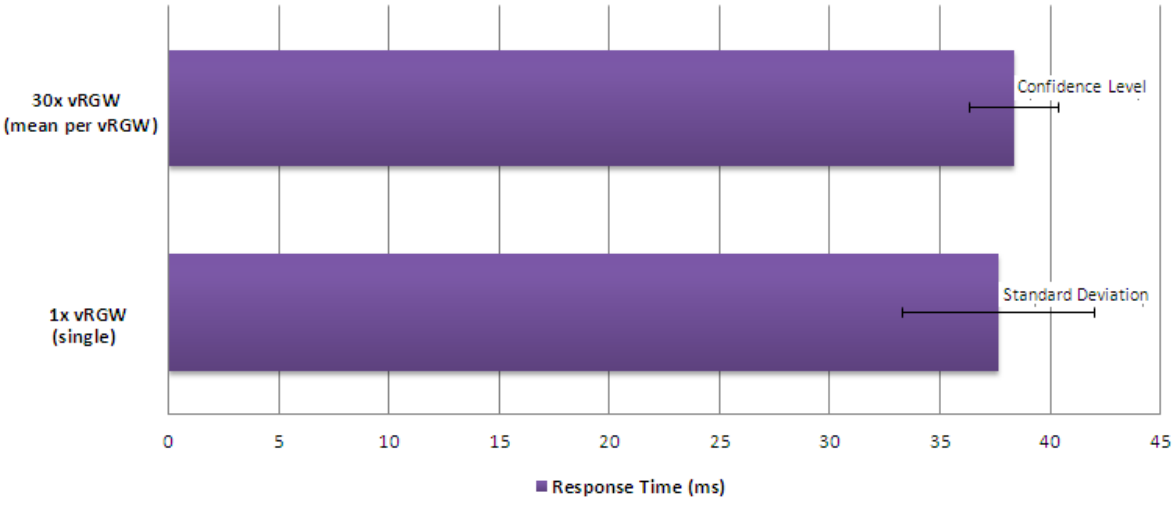


FIGURE 4.7: 1x vRGW (single) *vs.* the mean of 30x vRGW (bulk).

The chart from Figure 4.8 on the other hand shows the mean cumulative response time for the 30x vRGW bulk operation and the observed standard deviation on twelve different experiences.

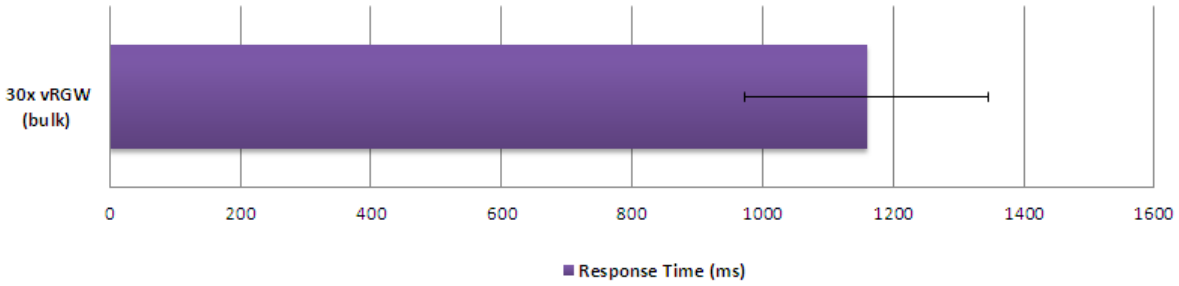


Figure 4.8: 30x vRGW (bulk) cumulative response time.

The statistical study on response times have shown that the proxy worked extremely fast even during bulk operation requests. The difference against a standalone system call request was minimal as observed in Figure 4.7.

When performing during bulk operations, the overall response times have also proven to be quite accurate with a 99% confidence level of 2 milliseconds despite the registered interval of about 107 ms.

To further asses our system, we have also decided to capture traffic for the operation requests. The charts from Figure 4.9 and Figure 4.10 represent the registered packet volume and byte volume between the ACS, the extensible CWMP agents and each vRGW respectively.

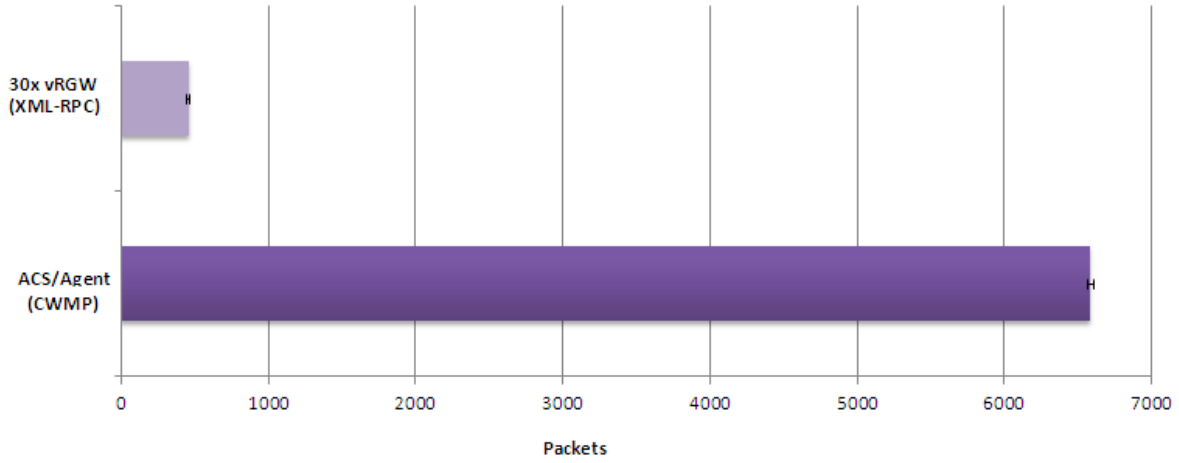


Figure 4.9: Registered packet volume on a CWMP request.

The traffic measurements have demonstrated the following:

- The generated traffic is quite significant between the ACS and the extensible CWMP agents (CWMP protocol exchanges only), being it a bulk operation with many vRGWs involved or only one.
- Each single call from the CWMP subagent to each one of the 30 vRGWs, represented only 1620 bytes average, with 12 bytes corresponding to a 95% confidence level for the mean.

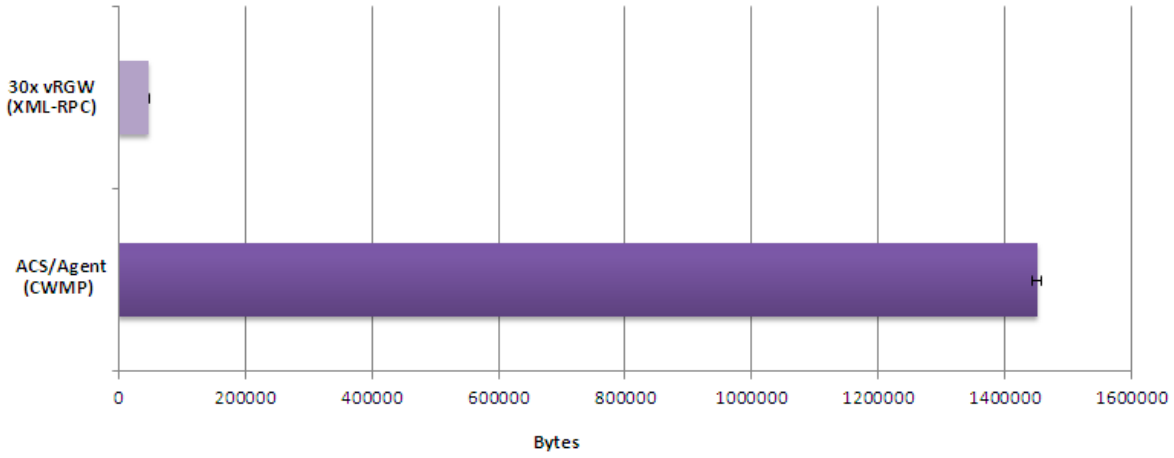


FIGURE 4.10: Registered byte volume on a CWMP request.

The presented traffic reads have led us to state that CWMP is clearly an inefficient protocol with lots of room for improvement, concerning the management of vRGWs. The specific arrangement of several vRGW instances *per* virtualization node is in part the reason for this inefficiency, because the concept of batch requests does not exist currently.

During our experiments we have observed CWMP protocol exchanges between the ACS and the extensible CWMP agent being multiplied by the number of RGWs involved in the batch request, where only one CWMP protocol exchange would have been enough to complete the batch successfully. Therefore, it is clear to us that batch requests should be addressed by CWMP frameworks in future protocol developments.

In order to complete the testbed assessment, we have also made batch reads using the *top*[32] utility every second during five minutes. Our goal was to collect the daemon CPU and memory utilization footprint at each vRGW.

Our read results showed that the "CWMP Proxy" used a constant amount of 6.4 Mbytes RAM (20% in our testbed) at each vRGW while CPU was 0% at all times. This can be easily explained, basically the daemon itself just stands waiting for call request (no CPU utilization), CPU time cycles depend almost solely on the effort needed to accomplish each system call request independently.

Regarding the RAM reservation amount, basically the "CWMP proxy" was using most of it to handle the exchange of XML messages and their associated parsing variables.

The type of CWMP operation requests issued on the ACS to further evaluate our implementation were the following:

- Reboot
- Package Update
- Package Install
- Package Remove
- Package Upgrade
- Add new firewall rule
- Reconfigure Ethernet LAN interface IP address

All the aforementioned requests completed successfully in matters of a few milliseconds. The statistical study presented here had the last request from the list as reference, although all the other requests can be marked out by the *Reconfigure Ethernet LAN interface IP address* request.

During our experiments, the CPU utilization percentage has remained constant in 0%. This basically proved that the XML-RPC service itself did not represent much of an effort for the operating system. We can guess that was because the daemon was only returning the system call return code back to the extensible CWMP agent which represented extremely short response messages with no impacts on CPU utilization.

5
Evolution of the Research

This chapter intends to show how the research has evolved over time.

5.1 Introduction

When the research activities have started back into September 2011 there was not an effective research plan promptly available. Everyone was still too stuck to the late edition of project S3P, the majority of the previous team members had left weeks before and the new team members have joined in to begin project virtuoso hence the course of things was initially somehow kind of a grey area.

Nevertheless, the main research goal was clear, we wanted to define a full-fledged virtualization architecture for remote gateways. Project SP3 had produced a significant amount of documentation that would be used as ground basis in future research activities. On its late edition, project S3P have proposed a management framework for CPEs introducing an extension to standard CWMP that would enable management of non compliant CWMP CPEs at customer premises.

In the meanwhile other documentation sources were identified as good starting point to begin defining our aimed virtualization architecture for RGWs.

From the beginning, there has not been an obligation to follow a rigid path to achieved our main goal and with time we have moved from documentation to start creating prototypes and defining what we believed to be the best validation metrics for them, always keeping track of our progress.

There has been a point in time where we have understood that our research was already enough to start writing papers and begin publishing our work.

The next section briefly describes the publications regarding this research.

5.2 Publications

To this day we have produced two paper articles:

1. An architecture for Virtualized Home Gateways (A.1)

   This paper has been finished in April and considered mature enough to be submitted to CNSM 2012. In the meantime we should have a paper notification by July 20, according to the General Chair of the conference.

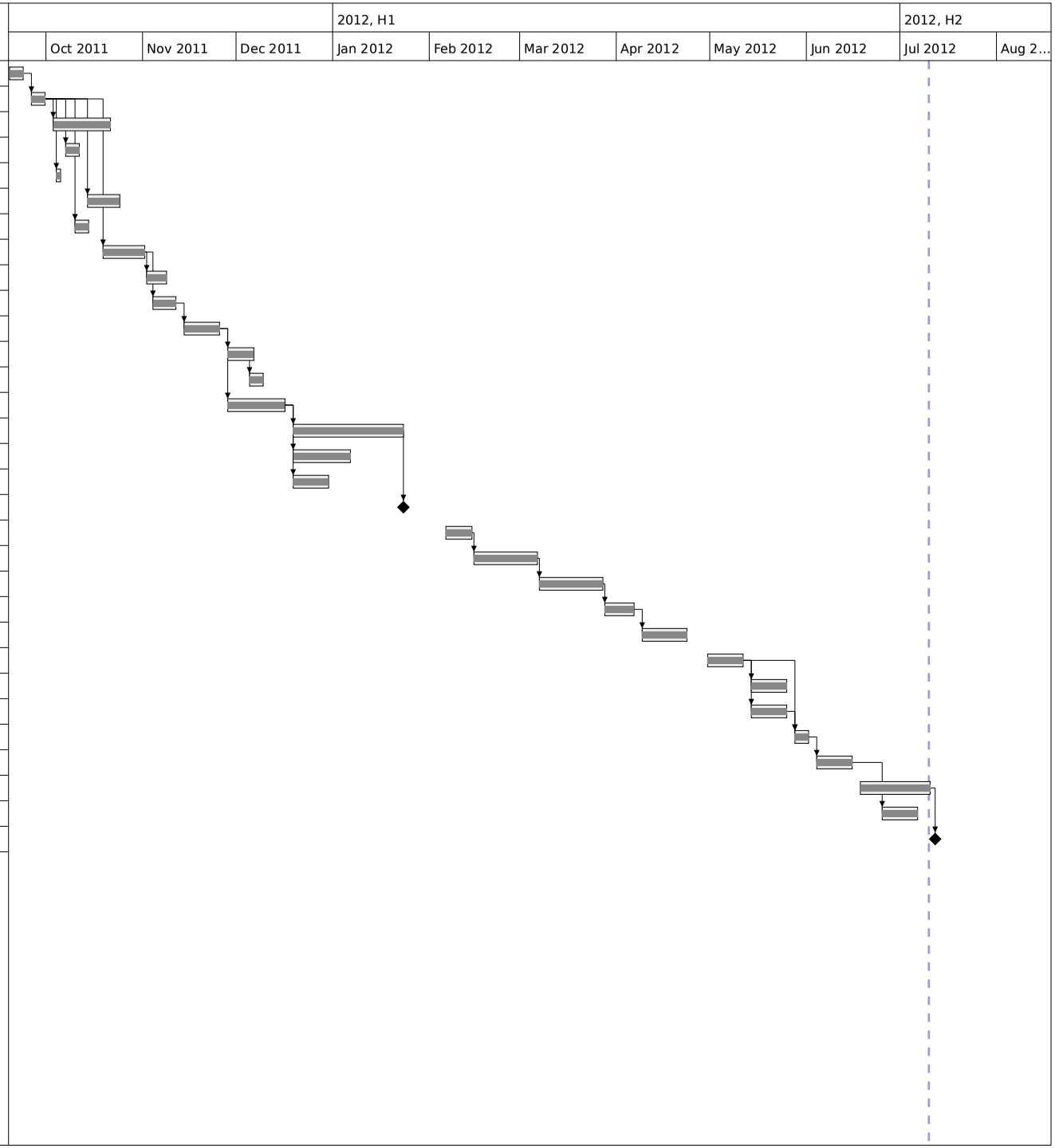
2. Management of Virtualized Home Gateways (A.2)

   This paper is still in it's early stage. Tough, the lab work is now completed and it will only be necessary to refactor the writing to the point where we think, it is ready to be submitted. According to our plans the paper should be ready to submission in the end of July.

5.3 Planning

The next page shows the gantt chart on how the research as evolved over the last 10 months.

Name	Work
Study Braodband Forum documentation	5d
Study published articles and papers	5d
Study existing TR-69 framework and available projects	15d
Study Eurescom virtualization proposals	3d
Perform some tests using HTML5, H.264 and WebM	2d
Create a dummy project using TR-69 java framework	7d
Study and propose a virtualization platform	5d
Paravirtualization tests with dd-WRT on KVM and QEMU	10d
Study and start PoC on dd-WRT mod-kit	5d
Assess openWRT as alternative for dd-WRT	6d
start PoC on openWRT (multiple instances)	10d
produce study on relevant metrics for future virtualization enhancements	7d
prepare thesis ToC proposal	5d
define startegy to inject traffic on testbed scenario	15d
start writting interim report	26d
investigate traffic sources and characterization studies	15d
understand how to fit current virtualization scenario on BB forum's virtual network model	10d
deliver interim report	
Start working on first paper	7d
Define better validation tests (virtualization)	15d
Investigate traffic injectors	15d
Execute new test battery using a different approach	8d
virtualization paper refinement	11d
Start research on managemet framework	10d
Develop extensible CWMP based prototype	10d
Define global management architecture	10d
Develop "CWMP Proxy" (XML-RPC Python daemon)	5d
Validate Management Prototype	10d
Start work on master thesis	17d
Write management paper	10d
Deliver master thesis	



5.4 Evolution of the Research

The management platform based on TR-69 has been developed previously to the writing of this report and to kick-start this project it was initially proposed the reading of documentation from Broadband Forum and also the research articles published by previous teams.

To become even more acquainted with CWMP and with the existing TR-69 framework, the next step was the writing of a dummy project where the java framework would be used. The dummy project was developed and resulted on an online radio provision solution. It consisted on provisioning the home user with a playlist of online radios that ideally could be listen on TV through the usage of an STB.

Continuing, virtualization has been finally picked and a new discussion was raised upon the subject of what virtualization platform would be better for the project. The first proposals pointed to XEN and KVM over other commercial or partial open-source counterparts like VMWare or VirtualBox. As for the virtual Residential Gateway OS, dd-WRT[35] has been initially picked up.

A Poof of Concept (PoC) has been set to assess if dd-wrt was virtualizable with KVM and libvirt. If dd-wrt proved to be virtualizable on KVM, the next step would be assessing scalability and performance on a scenario with multiple dd-wrt instances running on a unique hypervisor.

The PoC also wanted to understand if dd-wrt was flexible enough to be modified in order to add and remove software and services from it targeting mainly the future inclusion of a TR-69 daemon.

The PoC proved that dd-wrt was indeed virtualizable on KVM, however not very flexible and it would be an hard task to modify the base OS image, mainly because the available mod-kit was short for the required modifications.

After some experiments with the mod-kit it soon became clear that it was missing some important features such as modular components, easily changeable, installable and uninstallable. The current dd-wrt development is also slow and not fully available freely.

Nevertheless, KVM proved to be a good virtualization platform suiting all the needs. KVM was open source, and continuously developed and is supported by major open source companies such as Red Hat. There was also plenty of documentation available.

After the results from the first PoC, some research on dd-wrt alternatives has been started and almost immediately OpenWrt[23] showed up. A new PoC has been initiated to determine if OpenWrt was indeed suitable for the virtual Residential Gateway OS.

The results with OpenWrt were very positive and after a while it became clear that this was the right platform, based on the fact that it was completely open sourced and it had a large community driven support base. OpenWrt comes with a good usable package management system and it is very well documented. It also has a very nice SDK, making the task of porting existing *C* code humanly possible which can be seen as a big plus. It has been so decided to go along with OpenWrt.

The research work from February to July has been more focused in publishing our results. From February to April, we have prepared a more complete reference testbed and defined a rigorous test plan to validate our proposed architecture for virtual

residential gateways.

All the research work during this time, including lab experiments, has been used to write the first paper about the topic which in the meanwhile has been submitted to CNSM 2012 in April 26.

From the end of April until July we have decided to propose a CWMP based management framework. Starting with the extensible CWMP framework from project S3P we managed to create an interesting management solution that supports both, physical and virtual RGWs in a transparent way to service providers.

The results from this other research topic have been then used to write our second paper which we intent to submit by the end of July.

Finally, this thesis has also been started during the month of June, targeting the hard deadline of July 12 for delivery.

6
Conclusion and Future Work

This research proposes a full-fledged framework for virtualization of Residential Gateways which can introduce significant benefit to the service provider and to the triple play service subscriber. The service provider will benefit from lower costs that the implementation based on vRGW offers and from the possibility of introducing the new vRGW model while keeping the current deployed physical RGWs. On the other hand triple play service subscribers will also benefit from the faster introduction of new services on the network by chance cheaper or corresponding to broader offerings in terms of existing commercial packages.

The virtualization framework also includes a network architecture that uses existent key network technologies, such as MPLS to extend the subscriber home network to the service provider service private cloud. This approach allows that certain services may be re-structured to become transversal to groups of vRGWs. With this new concept in mind we have designed the new management framework for vRGWs that follows what we called the hybrid approach.

Regarding our implementation of the management framework, it also worth mentioning the possibility of management of both types of RGWs in a transparent way to the service provider.

An integrated management system is like the next step for massive vRGW introduction on triple play service offers because operators will not have to change their current network management systems neither have to change the way they do management on current physical RGW devices, which enables co-existence of both devices.

The presented management framework combines several already existent concepts and technologies such as the proxy concept[10], already available for physical gateways which can be easily transported to the virtualization node in a private cloud deployment scenario with vRGWs.

This makes all the sense to us because with the introduction of vRGWs, what is likely to be seen is the gradual decoupling of functionality turning the initially proposed

vertical segmentation for RGWs into an hybrid approach as already mentioned.

Furthermore, our "CWMP proxy" implementation in Python made the daemon easily portable to other device types which can be considered a plus in future developments, regarding virtualization of other CPEs.

Future work is of course the actual implementation of the proposed architecture on a real service provider network infrastructure using real equipment.

Moreover, the live migration of a vRGW is still an issue because even given the fact that live migration of a VM image is indeed possible, the network connectivity would not follow. Each vRGW has an aggregated C-VLAN that comes out from a larger VLAN trunk which imposes the definition of a manual process to live migrate a vRGW from one virtualization node to the other.

Regarding enhanced management of resources and services, other services beyond our management framework implementation, namely stateless services running on current RGWs, such as the intrusion detection system (IDS) could be thought leveraged to an higher level traversing several vRGWs and running on dedicated servers with larger resources. Once again this would contribute to our vision of an hybrid set of services.

Furthermore, still on the enhanced management framework, the optimization in CWMP batch operations would be a major target to future improvement. CWMP does not comply with the concept of batch/bulk operation and that is basically because current physical RGWs standing between the home LAN and the Access Network, typically handle one only request at a time. Introducing some minor changes into the CWMP data models would make batch operations possible. This is particularly relevant given that vRGWs are typically deployed in sets *per* various virtualization nodes in a private cloud deployment scenario. Thus it would be a nice improvement to the standard protocol being able to perform one unique batch request given a certain list of CPEs. From the assessment in our validation scenario (4.2.3), such an improvement would drastically reduce the traffic amount, preventing all the repetitions observed between the ACS and the master agent.

A
Publications

This appendix describes the setup procedure for the reference testbed described in Chapter 4, section 4.1.1 and depicted by Figure 4.1.

The necessary steps to have the full scenario up and running are enumerated and described in detail next:

A.1 An Architecture for Virtualized Home Gateways

An Architecture for Virtualized Home Gateways

Nuno Reis, Tiago Cruz, Paulo Simões,
Edmundo Monteiro
DEI-CISUC – University of Coimbra
Coimbra, Portugal

Fernando Bastos
Alexandre Laranjeira
PT Inovação
Aveiro, Portugal

Abstract— the convergence of technical advances in the field of virtualization has enabled the consolidation and scaling of resources in a cost-effective way, a trend that has also found its way into the telecommunication operator infrastructure foundations, spawning from data centers to networks alike. Starting from the core, the impact of these developments is gradually reaching towards the edge of the infrastructure and into the access network.

In this spirit, we foresee the opportunity of virtualizing a key device of modern broadband access networks: the Residential Gateway (RGW). Presently located on the customer premises, the RGW stands between the home network and the access network. It imposes a considerable cost for the operator (acquisition and operation) and constitutes a single point of failure for all the services offered to the residential customers – such as Internet access, VoIP, IPTV and Video-on-Demand.

In this paper we propose an architecture for virtualized Residential Gateways (vRGWs) which physically removes the RGW from the customer premises, moving it into the operator data center as a virtualized entity. This solution potentially reduces deployment, maintenance and operation costs, whilst improving overall flexibility, reliability and manageability – both for the access network infrastructure and for provided services.

Keywords— Cloud Computing, Virtualization, Home Networks

I. INTRODUCTION

Looking into today's access network deployments, we see that Fiber-To-The-Premises (FTTx) network topologies are becoming predominant, with current DSL based topologies being gradually replaced by optical fiber. This opens an opportunity window for operators to rethink how some of their service offers are delivered, in order to reduce costs and improve flexibility and manageability.

The concept of Residential Gateways (RGW), also designated as Home Gateways, has remained more or less unchanged for some time. The RGW is a physical device that stands on the customer premises, providing the interface between the home network and the operator's access network. RGWs handle local network services such as DNS, DHCP, NAT, routing, firewalling, and IEEE 802.11 [1] wireless connectivity. They also provide direct support for added-value services such as IPTV (IGMP proxying [2] and VCI/VLAN management [3]) and VoIP (SIP [4] gateways and/or analog terminal adapters). The RGW is a feature-rich embedded system that, in spite of its importance, represents a considerable burden for the operator:

- The RGW device is relatively expensive. This cost can be either subsidized by the operator or transferred to the customer, but one way or the other it represents a relevant share of the initial deployment costs.
- RGWs are relatively complex and, therefore, prone to hardware failures and/or to misconfiguration. Furthermore, due to their location, they constitute an unavoidable single point of failure.
- Due to the above-mentioned reasons, RGWs often require on-site maintenance. This represents a considerable burden for the operator, due to the involved logistics.
- Operator time to market is often dependent on the unit manufacturer to introduce new services to their subscribers, which is obviously a serious penalty to pay. This is aggravated by the subsequent need to remotely upgrade thousands or millions of devices (a cumbersome and error-prone operation).
- It is very difficult for the operator to keep a homogeneous set of RGWs, which affects manageability. Even if the operator adopts a single model from a single vendor, minor firmware and hardware revisions over time gradually compromise homogeneity. In extreme cases, operators were even forced to massively replace existing RGWs in order to support new services.

Considering this scenario, it would be attractive to consider alternative approaches able to overcome or at least mitigate these problems. It is obviously impossible to completely remove the RGW physical device from the customer premises: it will always be necessary to bridge the local network devices (computers, set-top-boxes, telephones, etc.) with the access network. Nevertheless, a very large part of the functions currently hosted by the physical RGW device can be moved closer to the operator's infrastructure, since the constraints that originally led to the hosting of such functions at the customers premises are gradually becoming less relevant:

- The bandwidth supported by the access network is no longer a limiting factor. Commodity FTTx offers easily support 100 Mbps and above.
- Recent advances in the field of virtualization and cloud computing make it possible to massively virtualize at least part of the functionality currently supported by the RGW, using a private cloud approach.
- Network technologies such as 802.1Q VLAN [5], VLAN stacking [5] and MPLS Pseudo-Wire (PS [6]) provide the counterpart support from the access network and core network side, allowing to "extend" each home network into

the operator infrastructure in a scalable and safe manner.

In this paper, we explore this idea of virtualizing RGWs, moving most of its functions to the operator infrastructure (using virtualization and private clouds) whilst keeping a drastically simplified device at the customer premises to support the remaining functions that, by their own nature, cannot be moved.

The rest of this paper is organized as follows. Section 2 discusses the motivation for virtualized residential gateways. The proposed approach is presented in Section 3 (network architecture) and Section 4 (virtualization of the RGW). Section 5 addresses validation. Section 6 discusses related work and Section 7 concludes the paper.

II. The Case for Virtualized RGWs

The notion of Virtual Residential Gateways (vRGW), as proposed in this paper, is a natural extension of the current trend towards Cloud-based services that is gradually extending the reach of the Cloud concept to the operator's network infrastructure.

Cloud computing has been made possible by a mixture of technological factors, such as the emergence of virtualization technologies and the widespread availability of broadband network access, enabling a multitude of new, cloud-sourced services, platforms and infrastructures. Specifically considering the infrastructure of telecommunications operators, Cloud services are already widespread – private clouds currently support key services such as content provisioning, authentication, authorization, accounting and management, in order to enhance their scalability, availability and reliability.

So far, the physical access network infrastructure has remained relatively excluded from this trend, since many of its components strongly depend on location. Nevertheless, even here there is space for improvements, namely by decoupling hardware-dependent functionalities (which can not be moved) from software-based functionalities, moving the software-based functionalities to the data center in order to reduce costs and to improve availability and flexibility.

Furthermore, there are some cases where the location constraints are simply related with the "logical location" of the components, not its "physical location". In this case it might be possible to redesign the logical network in order to extend its reach up to the operator data center – thus supporting the virtualization of such components.

For the specific case of the vRGW, what we propose is a mix of both methods: to decouple hardware-based and software-based functionalities as much as possible, and also to extend the logical reach of the customer's "home network" to the data center so that it can include the virtualized RGW. This implies that, as already mentioned, there is a remaining device left at the customer's premises, mainly for simple bridging purposes. Fig. 1 illustrates this approach.

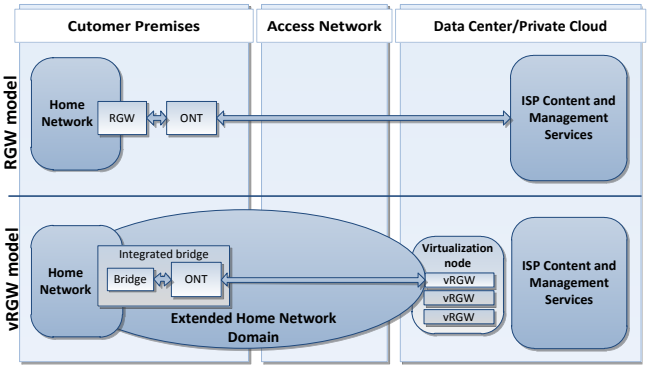


Fig. 1. Classic RGW vs. Virtualized RGW.

It should be mentioned that the proposed approach is not a mere transfer of a handful of services from the (physical) RGW to the operator infrastructure:

- First, in order to effectively overcome the limitations discussed in Section I, the remaining bridging device needs to be drastically simplified, in order to effectively reduce the associated capital expenditure (CAPEX) and operational expenditure (OPEX).
- Second, the access network infrastructure (and the data center) needs to accommodate thousands or millions of logical networks, in order to link the home network of each customer with its vRGW.
- Third, the virtualization technologies at the data center need to efficiently support a large number of vRGWs. Whilst commonplace virtualization tools can be used, it should be noted that the vRGW is different from the typical virtualized server, demanding less computing resources but more network performance (e.g. to support routing, NAT and firewalling).

The potential benefits of virtualizing the RGW, according to the proposed approach, are manifold:

- The operator would be able to lower CAPEX, since the cost of the remaining device would be much lower than the cost of a full-fledged RGW. Eventually, this device could be merged with the ONT (Optical Network Terminal), further reducing its costs.
- OPEX would also be improved, since the need for on-site maintenance would be reduced (less hardware and misconfiguration problems). This would also result in enhanced reliability.
- The aforementioned RGW updating problems no longer apply – updating an RGW is a mere manipulation of virtual machines handled at the data center. This also accelerates the introduction of new services.
- The operator becomes less dependent from hardware manufacturers (the virtual machine is, by nature, hardware independent).
- The heterogeneity of RGWs is no longer an issue. The operator can keep a unified image for all virtual machines (or, at most, a handful of images with different feature sets).
- The resources consumed by vRGWs can be efficiently

managed at the data center (e.g. suspending unused vRGW instances and dynamically adjusting the hosting hardware to effectively necessary resources), resulting in substantial savings of energy and hardware resources.

- Defective rRGW instances (e.g. due to software and misconfiguration problems) can be instantly restarted from fresh images, reducing downtime for the customer.

Next, we propose a framework for support of virtual RGWs. First, we discuss how to extend the home network, from a logical point-of-view, from the customer's premises to the ISP data center (Section 3). Then we discuss virtualization of the RGW at the data center and briefly address the bridging device that still needs to be left at the customer's premises (Section 4).

III. Proposed Network Architecture

Logically extending each customer's home network to the data center is a considerable challenge, due to the inherent scalability and manageability requirements. To the best of our knowledge, there are no proposals specifically addressing such a virtualization context. Nevertheless, two reference frameworks from the Broadband Forum [7] could be used for this purpose – even though they were not specifically developed with this objective in mind.

The Broadband Forum has defined Ethernet-Based Broadband Aggregation scenarios over DSL access networks (TR-101 [8]) and Gigabit Passive Optical Network (GPON) access networks (TR-156 [3]).

TR-101 provides an Ethernet-based architecture that has become a global standard for triple-play deployments for residential and business customers using DSL as access technology. Moreover, many of the specifications of TR-101 are not exclusive of DSL network elements.

Recognizing these benefits, many service providers planning GPON deployments were eager to use elements of the architecture and requirements provided by TR-101. This trend led to the release of TR-156, which strengthens this architecture for GPON scenarios, providing more detailed specifications.

A. Broadband Forum VLAN Aggregation Topologies

Considering GPON scenarios, the Optical Line Termination (OLT) and Optical Network Terminal (ONT) share the responsibility for Access Node VLAN requirements as specified by the Broadband Forum. TR-101 identifies three VLAN topologies:

- **Service VLAN (N:1)**. This topology defines a single VLAN per service, which carries traffic to all subscribers. Each new service requires a dedicated VLAN (S-VLAN). This fits quite well into the IPTV model, where multicast IPTV is delivered in the same VLAN for all subscribers.

  For GPON environments, TR-156 elaborates a bit further on this topology. The ONT adds an S-VID (S-VLAN ID) or translates an incoming S-VID tag for upstream traffic, so that there is always an S-VID between the aggregation node and the access nodes. The aggregation node will allow any upstream frames with an S-VID to pass-through. The downstream is essentially the opposite operation, with the aggregation node passing through the S-VID. The access nodes will remove or translate the tag and then forward frames to its associated U interface (ONT – Customer Premises). Fig. 2 illustrates this topology for both DSL and GPON.

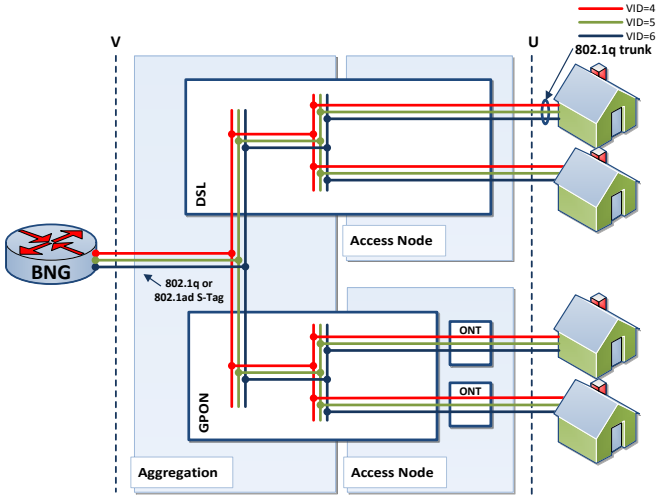


Fig. 2. VLAN per service model (adapted from [9]).

- **Customer VLAN (1:1)**. According to this topology, there is one VLAN per subscriber, which carries all its traffic. This topology implies a 1:1 mapping between services and Customer-VLANs (C-VLANs), trusting on the edge router to manage thousands of VLANs. However, this topology is not efficient with IPTV, since an IPTV stream being delivered to multiple subscribers would have to be carried across the network several times.

  Specifically considering GPON scenarios, the ONT maps each 1:1 VLAN into a unique U interface. Each U interface can map into one or more 1:1 VLANs. The ONT always adds a tag to untagged frames or translates an incoming Q-Tag in the upstream direction. Tag assignment at the V interface can follow two variations.

  The first variation corresponds to *single tagging*. For single-tagged VLANs at V, the ONT is provisioned to add an S-VID or translate an incoming tag into an S-VID, and the aggregation node passes through the tag. However, the 12-bit VLAN identifier only supports up to 4095 subscribers.

  Double tagging, using VLAN stacking [5] (initially standardized by 802.1ad [10] and later incorporated into the 802.1Q standard) makes it possible to increase VLAN capacity at the aggregation level. For double-tagged VLANs at V, the ONT is provisioned to assign a C-VID (C-VLAN ID) or translate an incoming tag into a C-VID, and the aggregation node adds the S-VID. This will allow operators to use a single VLAN to support subscribers that have multiple VLANs. The operator core network will carry traffic with double-tagged, stacked VLAN (802.1Q-in-Q) headers of multiple customers while maintaining the VLAN and Layer 2 protocol configurations of each subscriber and without impacting the traffic of other subscribers. The

stacked VLAN processing feature preserves VLAN IDs and keeps traffic in different subscriber VLANs segregated. This last case is depicted in Fig. 3.

The downstream is essentially the opposite operation, with the aggregation node removing an outer tag (if there is one present). The OLT will remove or translate the tag and then forward frames to its associated U interface.

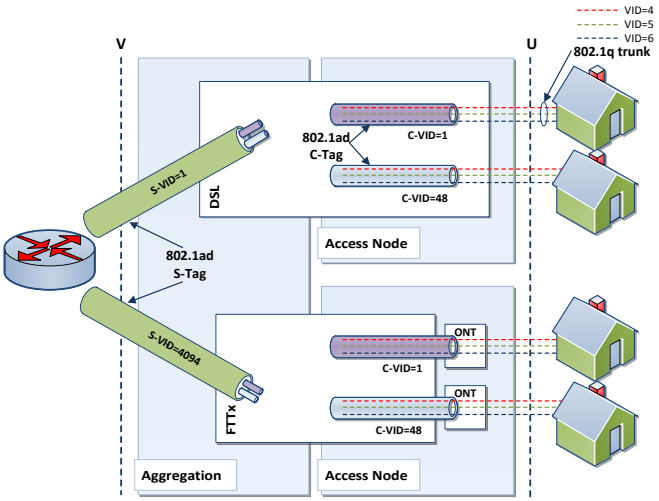


Fig. 3. VLAN per subscriber model (adapted from [9]).

- **Customer VLAN with Multicast VLAN**. This topology is a mixture, retaining the benefits from the two previous approaches. It purposes a shared multicast VLAN (M-VLAN) to carry specific services such as IPTV traffic [11] while the rest of the traffic is delivered using the 1:1 topology.

B. Proposed Architecture for Support of vRGW

Fig. 4 presents the proposed vRGW-enabling network architecture. According to this architecture, vRGW instances are hosted by Virtualization Nodes located at the operator's data centers (DC#1 and DC#2, in the case of Fig. 4), following a private cloud-sourcing model.

Each virtualization node is responsible for a set of vRGW instances. Each virtualization node is connected to the network by a mini VLAN trunk representing the I/O network traffic of every individual vRGW instance running in that specific virtualization node (i.e., the corresponding set of C-VLANs).

Each mini VLAN trunk is aggregated into a larger sub-trunk representing all the C-VLANs on a specific data center. This sub-trunk is then encapsulated into MPLS Pseudowires (PW) at one of the MPLS edge routers of the operator core network. Those PW are then transported to other edge routers, also known as Broadband Network Gateways (BNGs).

At the exiting BNGs, each PW is converted again into VLAN sub-trunks, each of them connected to the corresponding network access node.

At each individual access node, each VLAN sub-trunk is divided into smaller C-VLAN trunks carrying the traffic of each individual subscriber (separated by one VLAN for each service being delivered).

At the customer premises, on the ONT, each VLAN is untagged and mapped into a port in one unmanaged bridge with a wireless access point (the ONT and this bridge are not represented in Fig. 4, for sake of simplicity). At this point all the equipment inside the subscriber LAN can connect to the stripped-down bridge and obtain a valid IP address from the DHCP daemon running on the vRGW. From this point forward, the subscriber network usage experience is similar to what exists today with a physical RGW.

This network architecture allows extending the logical reach of the subscriber LAN to the vRGW hosted at the operator's data center using the technologies already in place. Moreover, the joint use of VLAN stacking, VLAN trunks and MPLS Pseudowires make it possible to handle, aggregate and encapsulate in a scalable manner the high number of VLANs required by thousands or millions of subscribers.

Considering the aforementioned Broadband Forum VLAN Aggregation mechanisms, the proposed architecture may fit both Customer VLAN (1:1) topologies and hybrid Customer VLAN with Multicast VLAN topologies.

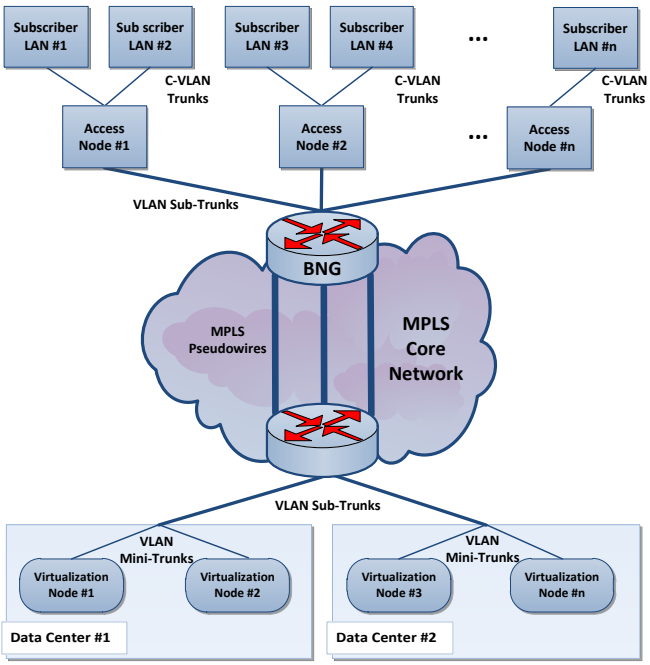


Fig. 4. vRGW-enabling network architecture.

IV. Virtualization of the RGW

RGWs typically consist of an embedded device based on specialized derivations of Unix, including services such as NAT routing, DHCP, firewalling, DNS, content filtering, QoS management, VoIP services and application-level gateways. Physical interfaces typically include Ethernet ports (home network, access network), a wireless access point, analog telephony adaptor (ATA) and, in some cases, USB ports.

As already mentioned, those physical interfaces cannot be completely removed, making it necessary to keep a bridging device at the customer premises. This device can be much

simpler than a full-fledged RGW, even if providing advanced interfaces such as an 802.11 wireless access point: the complex services and the most demanding network functionality – the interface between the home network and the Internet – are virtualized and moved to the data center. CAPEX and OPEX costs of such a device are expected to be much lower, especially if it becomes integrated into the already existing ONT.

From a virtualization environment point-of-view, the RGW imposes modest computing requirements (modest processors and a few hundred Mbytes of RAM), with the possible exception of the network interfaces between the access network and the home network. RGW consolidation by means of virtualization enables to further reduce their computing requirements.

Overall, this means that virtualization of RGWs can be performed using off-the-shelf virtualization platforms. Potential compatibility issues (due to embed hardware of RGWs) are also minimized by the fact that RGW firmware is usually based on Unix-derived stacks ported to several hardware platforms, including those based on the Intel x86 family.

A. Vertical Segmentation vs. Hybrid Approaches

In the specific scope of this paper, for sake of simplicity, it is assumed that RGW virtualization is achieved by simply migrating the RGW to the Virtualization Nodes, without extensive rearrangements of its architecture. Each vRGW remains isolated from the other vRGWs sharing the same private cloud.

Nevertheless, the concept of vRGW also opens the possibility of more advanced solutions, where some services previously handled by the RGW can become consolidated outside the boundary of each individual RGW, on a context of functional co-location (see Fig. 5). Application-level gateways such as SIP gateways and content caching mechanisms, for instance, could benefit from this aggregation to improve their manageability and functionality, as well as to reduce the vRGW footprint. Several RGW management functionalities could also be simplified if co-located at the level of the Virtualization Node, taking advantage of the new vRGW operating environment (remotely managing a physical RGW is different from managing local vRGW instances with support from an hypervisor).

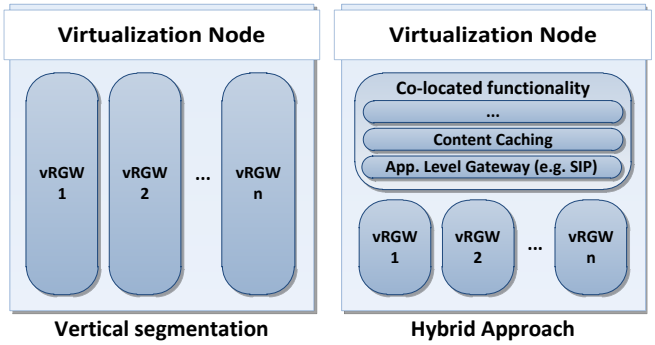


Fig. 5. Vertical Segmentation vs. Hybrid Segmentation.

B. Privacy and User-driven Customization

The proposed vRGW concept may raise concerns about user privacy and user-driven parameterization of the RGW.

Concerning user privacy, extending the home network up to the data center may lead to illegitimate probing of domestic network traffic (performed by the operator, at the data center). Nevertheless, this risk already happens with current setups, where the operator remotely controls the physical RGW (which might be remotely used for illegitimate probing). Either with vRGWs or physical RGWs, if the user does not trust the operator its only option is to hide the domestic network behind an additional NAT firewall.

The configuration of physical RGWs is typically shared between the customer (that uses a local web interface to define user-accessible parameters) and the operator (that uses CWMP [12] to remotely configure the other parameters). A similar approach might be followed for vRGWs, allowing the customer to access an operator-provided web interface to define user-accessible parameters of the vRGW (e.g. DHCP pools and 802.11 credentials). The configuration for each customer might be added to its profile and activated upon instantiation of the corresponding vRGW.

V. VALIDATION

A. Reference Testbed

In order to validate the proposed vRGW approach, we built the proof-of-concept testbed illustrated in Fig. 6, which emulates the data center environment and the Home Networks.

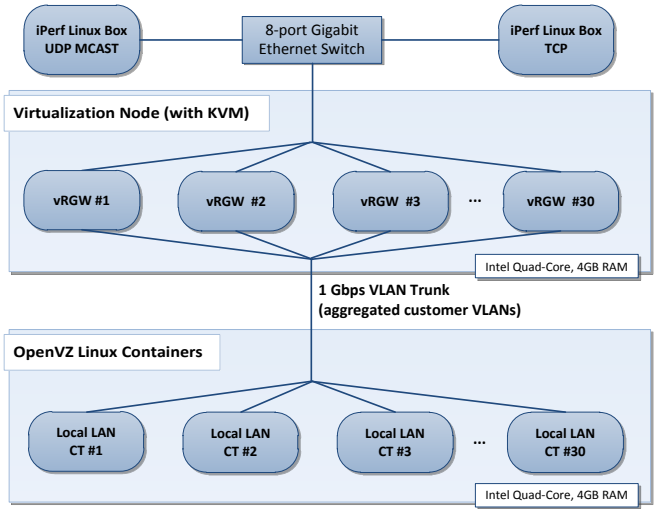


Fig. 5. vRGW virtualization testbed.

The virtualization node was based on an Intel server with 4GB of RAM, running CentOS Linux 6 x86 64 [13] with KVM (Kernel-based Virtual Machine [14]) as the virtualization hypervisor and *libvirt* [15] as management framework.

The interconnection between the Virtualization Node and the Home Networks was based on a 1Gbps Full Duplex Ethernet link, supporting one point-to-point VLAN trunk between the virtualization node and another CentOS 6 Linux x86 64 server running up to 30 OpenVZ Linux Containers

[16] (LXC) to emulate up to 30 independent subscribers (one per vRGW instance on the virtualization node). This VLAN Trunk aggregates the multiple C-VLANs involved.

The virtualization node was connected to a 8-Port Gigabit switch where two other Linux machines were running *iperf* [17], one of them configured as a UDP and UDP Multicast server source (to emulate VoIP and IPTV traffic, respectively) and the other as TCP generator (to emulate Internet connectivity and other general TCP traffic). The Quality of Experience Requirements for IPTV Services proposed by ITU for "Average Users" and "High Users" [18] (see Table I) were used to model *iperf*-generated UDP traffic (IPTV and VoIP services). The "TCP Internet" traffic generator was not capped, using the remaining available bandwidth.

It should noted that, despite the usage of UDP multicast for IPTV, a worst-case scenario was created by using "Customer VLAN (1:1)" mapping between the Virtualization Node and the Home Networks. This is also roughly representative of "Customer VLAN with Multicast VLAN" scenarios where each customer consumes distinct TV channels.

TABLE I. DOWNSTREAM BITRATES
FOR ALL-DIGITAL TRIPLE PLAY.

Service	Average User (Mbps)		High User (Mbps)	
Internet		5.0		10.0
Telephony		0.1		0.1
SDTV (MPEG-4)	2 channels	3.0	2 channels	3.0
HDTV (MPEG-4)	1 channel	8.0	2 channels	16.0
Total		**16.1**		**29.1**

B. Virtualization of the RGW

The presented proof-of-concept prototype uses OpenWrt Backfire x86 RGWs [19], virtualized using the KVM hypervisor and the *libvirt* management framework. Experimental measurements were collected using *virt-top* [20] for showing stats of virtualized domains.

OpenWrt was chosen for practical reasons, since it is one of the two most popular opensource RGW implementations. OpenWrt is a mature and well-organized project, easy to adapt for out purposes (selection and management of components, support for x86 architectures, support for customization). Besides, OpenWrt is in fact representative of a classical RGW (several commercial devices are actually based on OpenWrt).

KVM was selected because it is an opensource virtualization framework mature enough to be considered on pair with commercial platforms and not subject to legal benchmarking/disclosure constraints. It is compatible with management frameworks (such as Open Stack [21] and Open Nebula [22]) and can support large scale virtualization architectures, including management of live migration, load balancing, provisioning, virtualization node life cycle, high availability, power saving and storage. Those features are not pertinent in our small proof-of-concept prototype but might become relevant in future developments.

C. Experimental Results

In order to assess the capability of our testbed, we conducted measurements varying the number of vRGWs and considering ITU-defined requirements for "Average Users": 2 SDTV channels, 1 HDTV channel and 3 phone calls using the G.711a codec (which exceeds the 0.1 Mb/s specified by ITU for telephony). For "High Users" we considered 2 SDTV channels, 2 HDTV channels and 3 phone calls.

Fig. 6 presents the average throughput measured for each Home Network, with 1, 10, 15, 20, 25 and 30 vRGWs hosted by the Virtualization Node and the "Average User" scenario. Results show that the ITU reference requirements were easily surpassed, even with 30 vRGWs and the modest hardware used for the Virtualization Node.

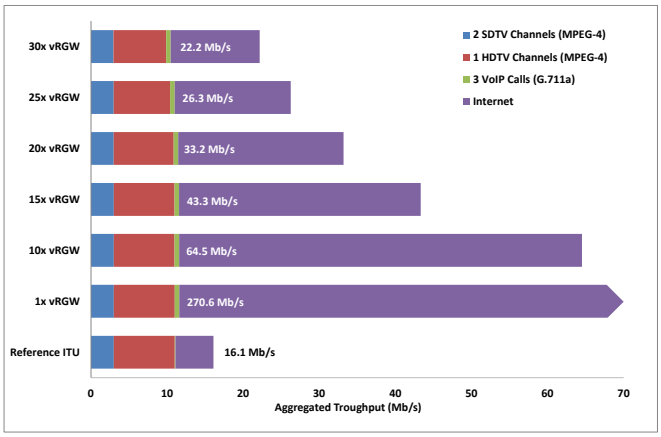


Fig. 6. Measured Throughput per vRGW ("Average User").

The measured average throughput per Home Network achieved for the "High User" profile is presented in Fig. 7. According to these results, our prototype should be able to support between 20 and 25 "High Users".

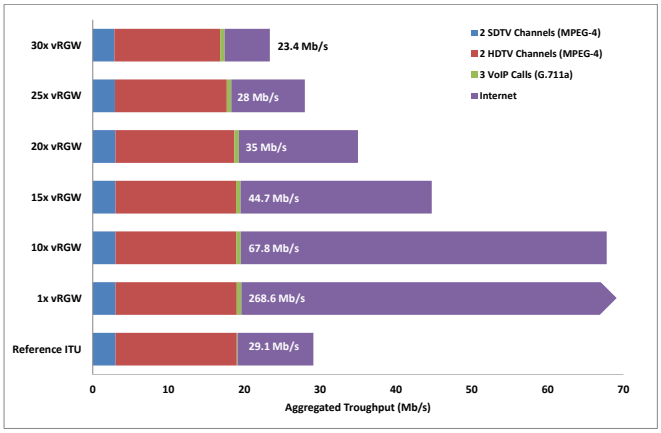


Fig. 7. Measured Throughput per vRGW ("High User").

We noticed small but noticeable packet loss for the UDP traffic in the most demanding scenarios (many vRGWs). The three worst cases were: 4.30% packet loss (30 vRGW and "High User" profile), 2.35% (25 vRGW, "High User") and 1.57% packet loss (30 vRGW, "Average User"). This can be justified by the fact that we used no traffic prioritization, an unlikely scenario in production environments. Furthermore, such packet loss ratios can be handled by the forward error correction (FEC) mechanisms employed by IPTV platforms without any noticeable degradation of quality [23-24].

During the experimental tests we also monitored the Virtualization Node, in terms of CPU load and memory usage.

Fig. 8 shows the total CPU load used by the hypervisor of the Virtualization Node (represented as a percentage of the maximum CPU capacity), for the "High User" scenario. Apart from the single vRGW case, the average CPU usage for 10, 15 and 30 vRGWs under load fits between 75% and 85%. This threshold could be explained by the fact that, besides running the virtualization hypervisor, the Virtualization Node also needs to handle the management of multiple VLANs with high load, running network I/O traffic on its VLAN trunk.

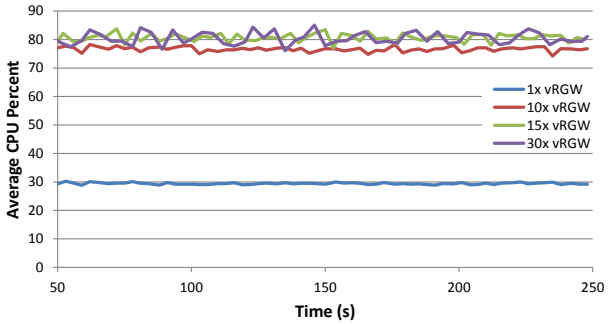


Fig. 8. CPU used by the Hypervisor ("High User").

Fig. 9 presents the average CPU load occupied by each individual vRGW (for the same "High User" scenario), represented as a percentage of the maximum CPU capacity of the Virtualization Node. Results are consistent with the hypervisor measurements of Fig. 8, showing no signs of remarkable overhead at the level of the hypervisor.

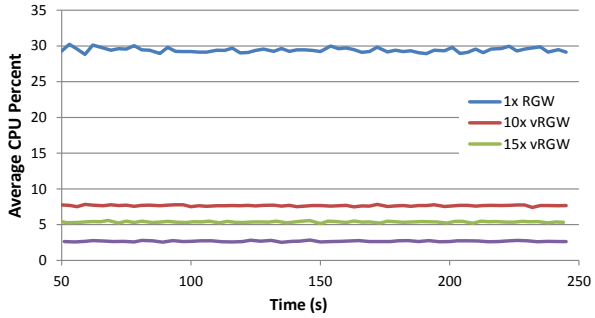


Fig. 9. Average CPU used per vRGW ("High User").

In terms of memory allocation, each vRGW OpenWrt instance was configured with 32MB of RAM, totaling up to 960MB (30x32MB) of allocated memory on the hypervisor. During the tests each vRGW consistently used around 18% of available memory (more or less 6MB), showing that RAM was not a key bottleneck in the specific system we used.

Finally, it should be stressed that the vRGW capacity per Virtualization Node, as hinted by presented measurements, is strongly capped by the modest hardware we used (old-generation Intel Q8400 2.66GHz, 4 GB of RAM, 2 entry-level Gb/s network interfaces). Commercial deployments are likely to achieve dramatic capacity improvements simply by careful specifying and tweaking critical hardware components, as well as by using the resource consolidation tools provided by data-center grade virtualization platforms.

VI. Related work

Router virtualization in general was addressed by a number of previous studies. Egi et al. [25-26] have researched the virtualization of routers within the Xen Hypervisor [27], using the *click* modular router [28] and XORP extensible router platforms [29]. This study focuses on the fair sharing of resources in scenarios where multiple users share the same virtualization host. Pisa et al. [30] evaluated the performance hit of migrating Xen virtual routers across different hosts.

The virtualization of security appliances is studied by [31], using a VMware platform to implement distributed firewalls. In the same scope of security appliances, [32] specifically focused on the virtualization-induced penalty at network level.

Zeng and Hao [33] addressed the virtualization of network appliances using the KVM framework, evaluating the virtualization overhead hit in terms of network performance and I/O path.

To the best of our knowledge, the only study that specifically addresses virtualization of residential gateways and its massive application in a broadband access network environment, apart from our own proposal, is the Eurescom Project P2055 [34]. This project provides a high-level discussion of three alternative scenarios to remove the physical RGW from the customer's premises, in order to optimize operator's CAPEX and OPEX:

- Pushing the RGW functionalities to the access nodes. This approach places packet processing near the subscribers and distributes the load across several geographically disperse nodes. However, it requires massive hardware upgrades of access nodes and fragments computing resources across the operators network – increasing complexity and costs.
- Integrated the RGW functionalities on the BNG (Broadband Network Gateway). This scenario keeps the network design unchanged (unlike the first one). However, the current BNG capabilities are not expected to support massive RGW virtualization, thus leading again to the need of hardware upgrade and fragmentation of computing resources across several BNGs.
- As a stand-alone network element (NE) located somewhere in the operator's metro network. This option has the advantage of not interfering with already deployed network elements but introduces a new hardware component on the network, with inherent costs and maintenance requirements.

When compared to these three alternatives, our proposal stands out for a number of reasons:

- RGW virtualization is based on tried and tested, hardware-independent virtualization technologies. This means they can be deployed using common data center resources and private cloud paradigms.
- By moving the vRGW to the data center, there is no fragmentation of the computing resources across the operators network (spreading complex and dedicated hardware components across access nodes requires more space, more energy, more on-site maintenance and more upgrades).

- The proposed network architecture implies no changes in the current network setups and can be progressively deployed. This results in an easier migration path, with the possible coexistence of physical and virtual RGWs.

VII. Conclusions and Future Work

In this paper we proposed a framework for virtualization of Residential Gateways, in order to reduce the operator's CAPEX and OPEX by taking advantage of the recent increase of available bandwidth and consolidation/virtualization technologies.

This framework entails a network architecture that takes advantage of already existing technologies to logically extend each customer's home network to the data center. This allows decoupling the functionalities currently supported by physical RGWs, keeping a reduced set of functionalities at the customer's premises (using a small bridge with minimum requirements) and moving the majority of functionalities to the operator's data center.

In a first approach, those functionalities may be maintained using isolated virtual RGW instances (vertical segmentation). Over time, part of these functionalities will likely be co-located and shared among vRGW instances, to improve functionality, efficiency and manageability (hybrid approach).

Future work will focus on validation of this proposal on a larger scale scenario (with real network infrastructures). The network architecture also requires further research, in order to support dynamic migration of vRGW across different data centers. The live migration of a vRGW image from data center to data center is trivial, but the network connectivity (the logical connection to the home network) does not follow dynamically, since each vRGW has an aggregated C-VLAN that comes from a larger VLAN sub-trunk (as depicted in Fig. 4). This makes it necessary to define an adequate process to move the corresponding C-VLANs from that original VLAN sub-trunk to a new one connecting to the new data center.

Other lines of future work relate to the enhanced management of resources and services. Several stateless services running on traditional RGWs could be leveraged to a higher level, traversing several vRGWs and running on dedicated servers with larger resources (according to the already mentioned hybrid segmentation approach).

VIII. Acknowledgements

This research work was partially funded by PT Inovação, in the context of the VIRTUOSO Project.

References

[1] IEEE 802.11 Working Group, "IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", June 2007.

[2] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002

[3] Broadband Forum, "Using GPON Access in the context of TR-101, TR-156", September 2010.

[4] J. Rosemberg et al., "SIP: Session Initiation Protocol", IETF RFC 3261, June 2002.

[5] IEEE 802.1 Working Group, "IEEE Std. 802.1Q-2011, Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", August 2011.

[6] S. Bryant, P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[7] Broadband Forum, http://www.broadband-forum.org.

[8] Broadband Forum, "Migration to Ethernet-Based Broadband Aggregation, TR-101 Issue 2", July 2011.

[9] G. Young, "Broadband Forum Overview with Focus on Next Generation Access", http://www.uknof.org.uk/uknof14/Young-Broadband Forum.pdf, September 2009.

[10] IEEE 802.1 Working Group, "IEEE Std. 802.1ad-2005, Virtual Bridged Local Area Networks Amendment 4: Provider Bridges", March 2006.

[11] V. Joseph, S. Mulugu, "Deploying Next Generation Multicast-Enabled Applications: Label Switched Multicast for MPLS VPNs, VPLS, and Wholesale Ethernet", Morgan-Kaufmann, ISBN 0123849233, July 2011.

[12] Broadband Forum, "TR-069 - CPE WAN Management Protocol specification v1.1, Amendment 4", July 2011.

[13] CentOS project, http://www.centos.org.

[14] A. Kivity, "KVM, One Year On", Presented at KVM Forum 2007, Tucson, USA August 2007.

[15] Libvirt project, http://libvirt.org.

[16] K. Kolyshin, "Virtualization in Linux", http://download.openvz.org/doc/openvz-intro.pdf, September 2006.

[17] Iperf, http://iperf.sourceforge.net.

[18] ITU, "Quality of Experience Requirements for IPTV Services, FG IPTV-DOC-0184", December 2007.

[19] OpenWrt project, https://openwrt.org.

[20] Virt-top, http://people.redhat.com/ rjones/virt-top.

[21] OpenStack project, http://www.openstack.org.

[22] Open Nebula project, http://opennebula.org.

[23] M. Ellis, D. Pezaros, C. Perkins, "Performance Analysis of AL-FEC for RTP-based Streaming Video Traffic to Residential Users", in Proc. of the 19th Int. Packet Video Workshop (PV), Munich, Germany, 2012.

[24] B. Nagel, "Demonstration of TVoIP services in a multimedia broadband enabled access network", MUSE Project Presentation at Broadband Europe 2007, Antwerp, December 2007.

[25] N. Egi, A. Greenhalgh et al., "Evaluating Xen for Router Virtualization" In Proc. of the International Workshop on Performance Modeling and Evaluation in Computer and Telecommunication Networks (PMECT) 2007, Honolulu, HI, USA, August 2007.

[26] N. Egi, A. Greenhalgh et al., "Designing a Platform for Flexible and Performant Virtual Routers on Commodity Hardware", in Proc. of the 16th GI/ITG Workshop on Overlay and Network Virtualization (NVWS) 2009, Kassel Germany, March 2009.

[27] P. Barham, B. Dragovic et al., "Xen and the Art of Virtualization", in Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP) 2003, New York, USA, October 2003.

[28] E. Kohler, R. Morris, B. Chen, J. Janotti, "The Click Modular Router", in Proc. of the 17th ACM Symposium on Operating Systems Principles (SOSP) 1999, Charleston SC, USA, December 1999.

[29] M. Handley, O. Hodson, and E. Kohler, "XORP: An Open Platform for Network Research", In Proc. of the ACM Workshop on Hot Topics in Networks (HOTNETS) 2002, New Jersey, USA, October 2002.

[30] P. Pisa et al., "Migrating Xen Virtual Routers with No Packet Loss", in Proc. of the First Workshop on Network Virtualizaton and Intelligence For Future Internet - WNetVirt'10, Búzios, Brazil, April 2010.

[31] D. Basak, R. Toshniwal, S. Maskalik, A. Sequeira, "Virtualizing networking and security in the cloud" in ACM SIGOPS Operating Systems Review, Volume 4, Issue 4, December 2010.

[32] A. Bazzi, Y. Onozato, "Feasibility Study of Security Virtual Appliances for Personal Computing", in IPSJ Journal of Information Processing, Vol 19, No 0, July 2011.

[33] S. Zeng, Q. Hao, "Network I/O Path Analysis in the Kernel-based Virtual Machine Environment through Tracing", in Proc. of 1st International Conference on Information Science and Engineering (ICISE) 2009, Nanjing, China, December 2009.

[34] D. Abgrall, "Virtual Home Gateway: How can Home Gateway virtualization be achieved?", Deliverable D1 of EURESCOM study P2055, September 2011.

A.2 Management of Virtualized Home Gateways

Management of Virtualized Home Gateways

Nuno Reis*, Tiago Cruz†, Paulo Simões†,
Edmundo Monteiro†
Department of Informatics Engineering
Faculty of Science and Technology
University of Coimbra
Coimbra, Portugal
*e-mail: nmreis@student.dei.uc.pt
†e-mail: {tjcruz,psimoes,edmundo}@dei.uc.pt

Fernando Bastos, Alexandre Laranjeira
PT Inovação
Aveiro, Portugal
e-mail: fbastos@ptinovacao.pt
e-mail: alexandre-s-laranjeira@ptinovacao.pt

***Abstract*—A new opportunity window opens with the existence of Residential Gateways (RGWs) in the form of virtualized instances, but then the question on how to handle management on this new form of network element emerges. Customer premises equipment Wan Management Protocol (CWMP) is currently a standard for management of physical RGWs and other complementary Customer Premises Equipments (CPEs), adopted by almost everyone who offer triple play services (internet, IPTV and VoIP), as such it stands as a natural candidate for management of the new virtual Residential Gateways (vRGWs).**

Current triple play service providers already have a large base of physical RGWs which they still need to support and manage. In order to do so, service providers had to deal with significant investment on their current CWMP capable management systems which they want to preserve or at max, extend.

In this paper we propose a simple architecture for management of virtual Residential Gateways with CWMP. The proposed solution meets that last requirement like a charm since it keeps the current network management systems unchanged and so we think that one of the main barriers, if not the only barrier, keeping virtual Residential Gateways from being massively deployed is now overcome thanks to integrated management of CPEs on a unique network management system.

***Index Terms*—CWMP, Management, Virtualization**

I. Introduction

Considering current access network deployments, it's notorious that fiber access is becoming largely available to triple play service subscribers with coper being slowly but continuously replaced by fiber. This fact enables several improvements to the way how services are currently delivered. This improvements are mostly related with management, maintenance and with the service nature itself resulting on more added value for both, the service provider and its subscriber.

The concept of Residential Gateways (RGW), also designated as Home Gateways, has remained more or less unchanged for some time. The RGW stands in the edge between the access network and the home network which implies on-site management more often than desired by service providers with all the costs that such interventions represent. Management of Residential Gateways (RGWs) have always been a complicated challenge for triple play service providers due to the sensitive positioning of this key network element on a triple play deployment.

The introduction of virtual Residential Gateways[1] (vRGWs) has eliminated or largely mitigated almost every major management problems identified in traditional triple play deployments with physical devices. Therefore services such as DNS, DHCP, NAT, routing, firewalling, and added-value service management such as IPTV or VoIP are now moved into the carrier's private cloud in the form of a vRGW. This type of RGW services are the cause of most of the reported incidents in current triple play deployment scenarios which often leads to on-site maintenance operations, so moving those into the carrier's private cloud enables faster responses to reported incidents with an overall increase on efficiency. Nevertheless it will always be necessary to bridge the local network devices (computers, set-top-boxes, telephones, etc.) with the access network and IEEE 802.11 wireless connectivity must also be assured locally but that's not a big issue since all the complex logic is now moved away from the customer premises.

The concept of a virtual gateway may have introduced significant advantages to triple play service providers by all the already mentioned reasons, however any service provider already selling triple play services won't want to throw away all the investment that was already done, replacing all the deployed physical RGWs by virtual network elements. Furthermore, every service provider would have by now a more or less complex network management system, they want to preserve and maintain. The majority part of the current network management systems sit on top of CWMP for management purposes, usual management tasks can range from a firmware update to group joins on a specific multicast stream to provide a new IPTV channel to the end user.

In this context it would be very interesting to provide integrated management of physical and virtual RGWs in a transparent way to operators. In this paper we explore the idea of integrated management of traditional physical RGWs and the new virtual RGWs using CWMP as the management protocol, giving total transparency to the operator on the execution of regular management and operation activities on both kinds of network elements.

The rest of this paper is organized as follows. Section III discusses the motivation for CWMP capable management of virtualized residential gateways. The proposed approach

is presented in Section IV (Proposed Approach). Section V addresses validation. Section VI discusses related work and Section VII concludes the paper.

II. CWMP

CWMP has first emerged as a solution for management of DSL modems, making secure auto-configurations, diagnostic routines, software/firmware management and monitoring tasks possible on such devices. On a later stage CWMP was gradually extended to support residential gateways and also a large panoply of other customer premises equipments.

CWMP is based on Simple Object Access Protocol (SOAP) web-services, it defines an API which stands on XML Remote Procedure Calls (XML-RPCs) and standardized data models for several types of CPEs. A secure layer can also be applied to this management operations through the usage of Secure Socket Layer (SSL) or Transport Layer Security (TLS). The CWMP remote management server is designated as the Auto-Configuration Server (ACS)[2].

III. Motivation

The introduction of this new concept of virtual Residential Gateways (RGWs) has brought a new challenge with it addressing manageability. Although virtual residential gateways could be managed by common cloud computing management frameworks like OpenStack or oVirt, the specifics of a network element such as the RGW probably require a different approach.

On the other hand looking at current network management systems used by triple play service providers to manage physical RGWs, we see either proprietary or CWMP based frameworks. In order to keep current management systems, service providers would be at most interested on extended functionally on their current systems to start adopting virtual residential gateways.

Following this thought it seems logical to seek for an integrated management framework that would manage both, physical and virtual RGWs in a transparent way.

CWMP is currently wide-spread by triple play service providers network infrastructures, as such, taking the key concepts out from the work on CWMP agents[2], we could envision a CWMP based management solution for virtual Residential Gateways.

The work on CWMP agents[2] offers a key advantage that is the possibility of modelling simple CWMP agents, clearly separating the CWMP specific routines which are usually common to every CPE from the set of management functionalities associated with each managed service within the CPE.

In order to be able to manage virtual RGWs that do not come with an embedded CWMP stack, we could think on a CWMP agent running on each virtualization node (hypervisor level) that would speak CWMP to the ACS and would execute system calls on the virtualized instances by means of an XML-RPC interface exposed by each virtual RGW for integration purposes.

An integrated management architecture for both physical and virtual RGW instances is presented bellow in Fig. 1.

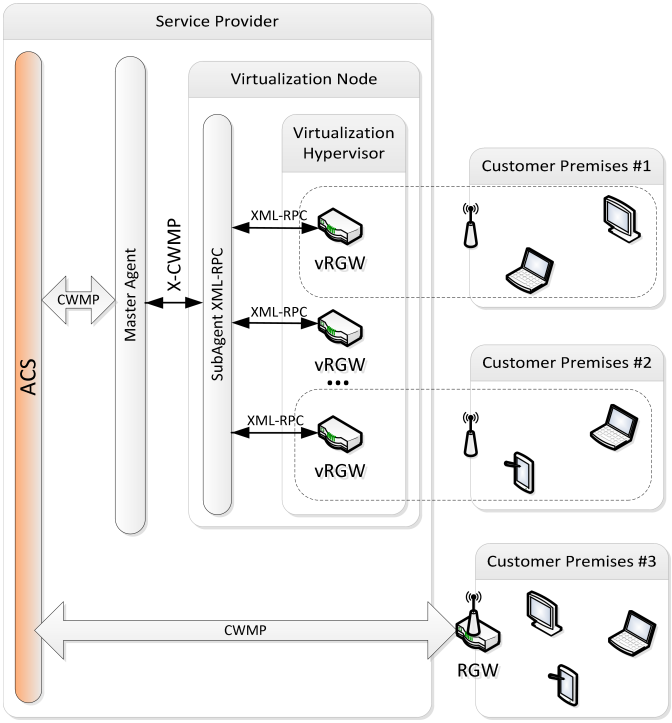


Fig. 1. CWMP based Integrated Management Architecture.

IV. Proposed Approach

The CWMP based integrated management architecture depicted in Fig. 1 shows the usage of what can be called a CWMP proxy which runs at each vRGW instance providing an XML-RPC interface for typical CWMP operations. This architecture could be better than others, because having a complete native CWMP stack on an embedded system could be extremely resource demanding and since we are using virtual RGWs it seems wise to decouple some stateless services such as a CWMP management stack to a different/higher/common level.

The proposed architecture keeps all the CWMP logic at the virtualization node level by means of an CWMP agent which cross talks with a simple XML-RPC [11] interface given by each vRGW, which in spite of everything, does not know anything about the CWMP protocol. It just acts as a common interface providing external system calls using XML-RPC protocol for the job. All the CWMP logic is kept at the CWMP agent that executes on the virtualization node and is basically divided in to layers. One layer speaks CWMP protocol to the ACS, while a second one speaks XML-RPC with the CWMP proxy to execute system calls on the vRGW instance.

The logical trend within vRGW deployments is to go for an hybrid approach leaving the vertical segmentation of physical RGW devices behind.

The proxy concept[4] also makes the overall solution extremely flexible to be adapted to other devices while keeping the CWMP complex logic all in one place. It worth mentioning that developing for embedded systems could be quite painfull to normal programmers due to resource limitations so common on this type of systems. Moreover, cross compiling existent code proves to be a daunting task, sometimes more painfull than developing everything from the scratch.

Finally looking at the proposed picture we can see that traditional management of physical RGWs keeps transparent to the operator and the vRGW addition is seen as if it was another physical device.

V. Validation

A. Experimental Testbed

In order to validate our integrated CWMP management framework, the testbed shown in Fig. 2 has been built to emulate the vRGW management operations only, although CWMP management of physical devices could be easily achieved by just connecting the ACS to such a RGW device.

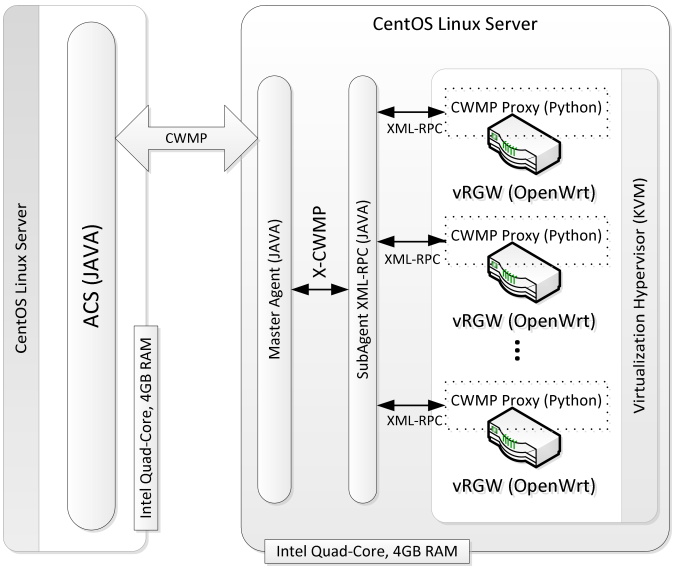


Fig. 2. CWMP based vRGW management testbed.

The virtualization node was based on an Intel Quad-Core server with 4GB of RAM, running CentOS[6] Linux 6 x86_64 with KVM (Kernel-based Virtual Machine) as the virtualization hypervisor.

The ACS ran on another CentOS 6 Linux 6 x86_64 server with the same hardware characteristics. The ACS consisted on a JAVA application executing as a standalone application on a simple java virtual machine (JVM), although it could be made available on a java application server (AS) such as JBoss[13] or Glassfish[14].

At the virtualization node we had another standalone java application running as a CWMP agent which implemented a simple CWMP data model for management of OpenWrt[16] vRGW instances.

At the KVM[15] hypervisor we have setup a group of 30 OpenWrt vRGW instances. One CPU core and 32MB of RAM have been given to each running instance.

At each OpenWrt vRGW we also had a python daemon listening to system call requests coming from the CWMP agent on the virtualization node. The connection between CWMP agent and CWMP proxy was managed by XML-RPC protocol. As already stated the CWMP proxy wasn't more than a simple XML-RPC server listening for system call requests on the OpenWrt operating system.

B. CWMP Proxy Concept

The presented proof-of-concept prototype uses OpenWrt Backfire x86 RGWs, virtualized using the KVM hypervisor and the libvirt[17] management framework.

Experimental measurements were collected using *top* for registering the python CWMP proxy daemon CPU and memory percentage utilization footprint at each vRGW instance and *tcpdump*[9] on the virtualization node to capture traffic between the ACS server, the virtualization node and each running vRGW according with the scenario from Fig. 2.

Moreover, in order to register typical CWMP operation durations, both the ACS server and the CWMP agent, both JAVA applications, were customized with a nanosecond timer for operation duration measurements.

OpenWrt was chosen for practical reasons, since it is one of the two most popular opensource RGW implementations. OpenWrt is a mature and well-organized project, easy to adapt for out purposes (selection and management of components, support for x86 architectures, support for customization). Besides, OpenWrt is in fact representative of a classical RGW (several commercial devices are actually based on OpenWrt).

KVM was selected because it is an opensource virtualization framework mature enough to be considered on pair with commercial platforms and not subject to legal benchmarking/disclosure constraints. It is compatible with cloud computing management frameworks (such as oVirt[12] and Open Stack[7]) and can support large scale virtualization architectures, including management of live migration, load balancing, provisioning, virtualization node life cycle, high availability, power saving and storage. Those features are not pertinent in our small proof-of-concept prototype but might become relevant in future developments.

On a first approach we could think on developing a native CWMP stack for OpenWrt, although it might seem a logical step, porting code to embedded systems such as OpenWrt is not an easy task and requires lots of effort. If we still didn't have any CWMP stack available, developing a new one from the scratch could take into account embedded systems, however we already had a java CWMP stack available to start with which of course was not available for OpenWrt or any other embedded system.

Looking into our options we came across with an idea of using a well known and scalable scripting language to create a proxy for a unique java based CWMP agent running at each virtualization node which would implement all the CWMP

logic there and would then cross connect with an XML-RPC interface running at each OpenWrt that would accept any kind of system call.

The proposed CWMP proxy for CWMP management of vRGWs also emerges from some noted facts:

- Both OpenWrt and libvirt already offer good management solutions.
- OpenWrt offers a great package management solution similar to what's currently found in desktop Linux distributions.
- On the other hand libvirt provides a great low level management interface to the KVM virtualization hypervisor.

Hereupon the logical question is: How can we take all this into account to develop CWMP management support?

The obvious answer was, with a simple XML-RPC proxy service.

Since Python[10] was already well-supported within OpenWrt, this scripting language was chosen for the job and it proved to be a wise choice as the results presented here can tell.

Architecturally speaking this CWMP proxy approach would easily adapt to any other kind of device which also makes it quite flexible.

C. Experimental Results

In order to assess the capability of our testbed, we conducted measurements varying the number of vRGWs from only one to thirty while registering operation response times to the request for changing the LAN interface address:

- ifconfig br-lan 192.168.255.1

Table I that follows bellow shows the descriptive statistics for the described bulk operation involving thirty vRGWs and twelve experiences.

The chart from Fig. 3 shows execution response times for a single request on one vRGW alone *vs.* the average response time *per* vRGW while performing a bulk operation with thirty vRGWs involved. The error bars on the same chart represent the 99% level of confidence for the mean on the 30x vRGW bulk request operation, while standard deviation is used for the single request.

The reason not to use standard deviation in both cases is because standard deviation is a dispersion measurement with little resistance, mainly because it is very sensitive to the influence of too large or too small values and as seen in Table I the maximum and minimum values in the sample were 124.832 (ms) and 18.314 (ms) respectively, which results in a standard deviation value of 14.922(ms). However calculating the 99% level of confidence for the mean we have gotten only 2.037 (ms). Thus in this particular case the 99% level of confidence for the mean was a better representation of reality.

Moreover, it also doesn't make sense to use the level of confidence for the mean in both cases, because the sample for the 1x vRGW single request was too small and bellow thirty, which is a common ground reference value for the calculation of the level of confidence for the mean.

TABLE I
EXECUTION RESPONSE TIMES DESCRIPTIVE STATISTICS.

	Time (ms)
Mean	38.386
Standard Error	0.786
Median	34.210
Mode	27.151
Standard Deviation	14.922
Sample Variance	222.664
Interval	106.518
Minimum	18.314
Maximum	124.832
Count	360.000
Confidence Level (95%)	1.547
Confidence Level (99%)	2.037

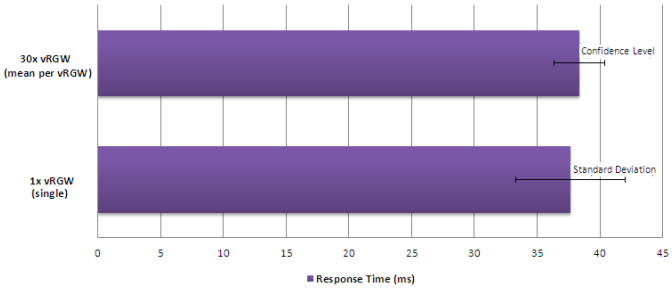


Fig. 3. 1x vRGW (single) *vs.* the mean of 30x vRGW (bulk).

The chart from Fig. 4 on the other hand shows the mean cumulative response time for the 30x vRGW bulk operation and the observed standard deviation on twelve different experiences.

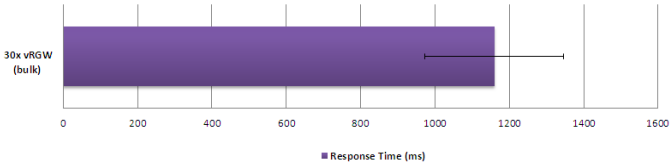


Fig. 4. 30x vRGW (bulk) cumulative response time.

The statistical study on response times have shown that the proxy worked extremely fast even during bulk operation requests. The difference against a standalone system call request was minimal as observed in Fig. 3.

When performing during bulk operations, the overall response times have also proven to be quite accurate with a 99% confidence level of 2 milliseconds despite the registered interval of about 107 ms.

To further asses our system, we have also decided to capture traffic for the operation requests. The charts from Fig. 5 and Fig. 6 represent the registered packet volume and byte volume between the ACS, the extensible CWMP agents and each vRGW respectively.

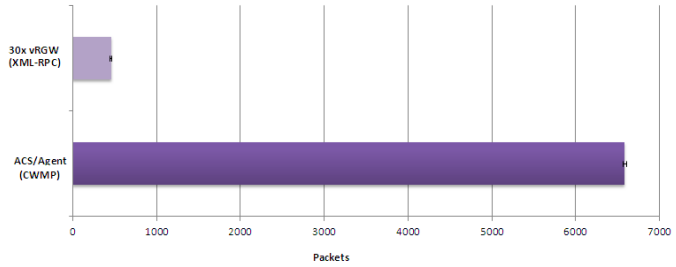


Fig. 5. Registered packet volume on a CWMP request.

The traffic measurements have demonstrated the following:

- The generated traffic is quite significant between the ACS and the extensible CWMP agents (CWMP protocol exchanges only), being it a bulk operation with many vRGWs involved or only one.
- Each single call from the CWMP subagent to each one of the 30 vRGWs, represented only 1620 bytes average, with 12 bytes corresponding to a 95% confidence level for the mean.

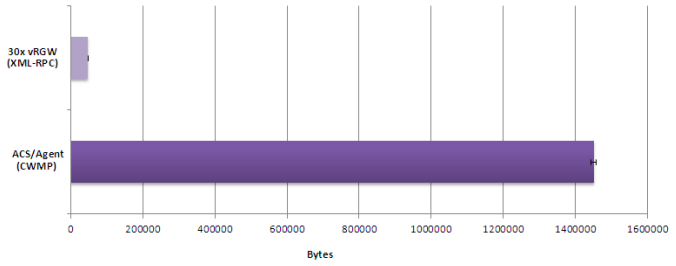


Fig. 6. Registered byte volume on a CWMP request.

The presented traffic reads have led us to state that CWMP is clearly an inefficient protocol with lots of room for improvement, concerning the management of vRGWs. The specific arrangement of several vRGW instances *per* virtualization node is in part the reason for this inefficiency, because the concept of batch requests does not exist currently.

During our experiments we have observed CWMP protocol exchanges between the ACS and the extensible CWMP agent being multiplied by the number of RGWs involved in the batch request, where only one CWMP protocol exchange would have been enough to complete the batch successfully. Therefore, it is clear to us that batch requests should be addressed by CWMP frameworks in future protocol developments.

In order to complete the testbed assessment, we have also made batch reads using the *top*[8] utility every second during five minutes. Our goal was to collect the daemon CPU and memory utilization footprint at each vRGW.

Our read results showed that the "CWMP Proxy" used a constant amount of 6.4 Mbytes RAM (20% in our testbed) at each vRGW while CPU was 0% at all times. This can be easily explained, basically the daemon itself just stands waiting for call request (no CPU utilization), CPU time cycles depend almost solely on the effort needed to accomplish each system call request independently.

Regarding the RAM reservation amount, basically the "CWMP proxy" was using most of it to handle the exchange of XML messages and their associated parsing variables.

The type of CWMP operation requests issued on the ACS to further evaluate our implementation were the following:

- Reboot
- Package Update
- Package Install
- Package Remove
- Package Upgrade
- Add new firewall rule
- Reconfigure Ethernet LAN interface IP address

All the aforementioned requests completed successfully in matters of a few milliseconds. The statistical study presented here had the last request from the list as reference, although all the other requests can be marked out by the *Reconfigure Ethernet LAN interface IP address* request.

During our experiments, the CPU utilization percentage has remained constant in 0%. This basically proved that the XML-RPC service itself did not represent much of an effort for the operating system. We can guess that was because the daemon was only returning the system call return code back to the extensible CWMP agent which represented extremely short response messages with no impacts on CPU utilization.

VI. Related Work

To the best of our knowledge, this CWMP based integrated management framework is the first one to be proposed that addressed at the same time, management of physical and virtual RGWs in a transparent way. However the general concepts on CWMP management for physical RGWs came from the work of Cruz, T. et al.[2] with their proposal for an X-CWMP agent extension framework for CWMP.

The same author also have a concrete proposal[3] for CWMP based management using decoupled CWMP agents, on Off-the-Shelf SIP Phones in Domestic and SOHO Environments which better demonstrates how to use the X-CWMP agent extension framework.

CWMP Management for home network devices is also covered by several technical reports from the broadband forum, namely TR-111[5]. TR-69[4] on their latest amendment 4 also addresses some of the concepts used on our CWMP based integrated management framework, namely annexes F and J.

The notion of device proxying based on CWMP-compliant gateways is also not novel. TR-69 Annex J covers CWMP Proxy Management and offers some interesting approaches regarding the CWMP proxy concept adopted by our implementation.

The broadband forum idealized the proxy concept for gateways (physical RGWs) but when vRGWs come into the equation this concept clearly makes sense when adapted for to virtualization nodes on a private cloud deployment scenario such as on our proposal.

TR-69 annex J still addresses the requirements for the data model and how to handle the proxied device availability.

VII. Conclusions And Future Work

In this paper we have proposed a CWMP based integrated management framework which addressed management of both physical and virtual RGWs in a transparent fashion.

The integrated management framework is like the next step for massive vRGW introduction on triple play service offers because operators won't have to change their current network management systems neither have to change the way they do management on current physical RGW devices, which enables co-existence of both devices.

This framework combines several already existent concepts and technologies such as the proxy concept[4] already available for physical gateways which could be easily transported to the virtualization node in a private cloud deployment scenario with vRGWs. This makes all the sense to us because with the introduction of vRGWs[1] what is likely to be seen is the gradual decoupling of functionality turning the initially proposed vertical segmentation into an hybrid approach.

Furthermore the results of our implementation of an XML-RPC interface acting as a CWMP proxy proved to be quite flexible and also extremely performant. The python implementation for OpenWrt had a constant reservation of 20% of the available RAM, 32MB in our testbed scenario, which means that running as daemon the CWMP proxy makes usage of only 6.4MB of RAM and replies to requests in matters of a few milliseconds to operation requests like a package upgrade, a package remove or a firewall adjustment.

The CPU utilization measured was always zero and only dependent on the concrete request effort to execute.

The chose of python also made the daemon easily portable to other device types which could be handy in future developments regarding virtualization of CPEs.

A. Future Work

Future work will focus on optimizing CWMP batch operations. CWMP doesn't comply with the concept of batch/bulk operation, that's basically because current physical RGWs stand between the home LAN and the Access Network and typically this devices only handle one request at a time.

Introducing some minor changes into the CWMP data models would make batch operations possible. This is particularly relevant given that vRGWs are typically deployed in sets per various virtualization nodes in a private cloud deployment scenario. Thus it would be a nice improvement if we could tell the ACS to perform a specific task on a virtualization node CWMP agent only once, instead of telling each co-located vRGW to perform the same task repeatedly until every vRGW on a virtualization node is covered.

From the assessment in our validation scenario, such an improvement would drastically reduce the traffic between the ACS and the CWMP agent on the virtualization nodes.

Acknowledgment

This research work was partially funded by PT Inovação, in the context of the VIRTUOSO Project.

References

[1] N. Reis et al., An Architecture for Virtualized Home Gateways, CNSM 2012.
[2] T. Cruz et.al, CWMP Extensions for Enhanced Management of Domestic Network Services, LCN 2010.
[3] T. Cruz et.al, How to Provision and Manage Off-the-Shelf SIP Phones in Domestic and SOHO Environments, LCN 2010.
[4] Broadband Forum, "TR-069 - CPE WAN Management Protocol specification v1.3, Amendment 4", July 2011.
[5] Broadband Forum, "Applying TR-069 to Remote Management of Home Networking Devices (TR-111)", December 2005.
[6] CentOS project, http://www.centos.org.
[7] OpenStack project, http://www.openstack.org.
[8] Unix top project, http://www.unixtop.org.
[9] Tcpdump project, http://www.tcpdump.org.
[10] Python Programming Language, http://www.python.org.
[11] Apache XML-RPC Java implementation, http://ws.apache.org/xmlrpc.
[12] oVirt project, http://www.ovirt.org.
[13] JBoss Application Server, http://www.jboss.org.
[14] Glassfish Application Server, http://glassfish.java.net.
[15] KVM project, http://www.linux-kvm.org.
[16] OpenWrt project, https://openwrt.org.
[17] Libvirt project, http://libvirt.org.

B

RGW Virtualization Testbed Setup

This appendix describes the setup procedure for the reference testbed described in Chapter 4, section 4.1.1 and depicted by Figure 4.1.

The necessary steps to have the full scenario up and running are enumerated and described in detail next:

1. Setup network in both Linux Servers:

   Create bridges

   Create pseudo-interfaces

   Create VLANs.

   This task was accomplished by using a simple bash script to automatize the procedure for any number of instances. The bash script is shown bellow in Listing B.1.

Listing B.1: bash script to automatize VLAN handling.

```
#!/bin/bash

NET_IF=eth1

ifconfig $NET_IF 0.0.0.0 mtu 1496 up

for i in $(seq 1 1 30)
do
        tunctl -u user
        brctl addbr br${i}
        if [ ${#i} -eq 1 ]
        then
                vconfig add $NET_IF 10${i}
                ifconfig "${NET_IF}.10${i}" 0.0.0.0 mtu 1496 up
                brctl addif br${i} "${NET_IF}.10${i}"
                brctl addif br${i} tap${i}
                ifconfig tap${i} 0.0.0.0 up
                ifconfig br${i} 0.0.0.0 up
        else
                vconfig add $NET_IF 1${i}
                ifconfig "${NET_IF}.1${i}" 0.0.0.0 mtu 1496 up
                brctl addif br${i} "${NET_IF}.1${i}"
                brctl addif br${i} tap${i}
                ifconfig tap${i} 0.0.0.0 up
                ifconfig br${i} 0.0.0.0 up
        fi
done
```

   This script is quite basic, the *NET_IF* variable only needs to be adjusted to use the appropriate Ethernet interface to be used on the VLAN trunk.

   The *for* loop is adapted to match any number of VLANs/vRGWs. This testbed was built as described for 30 vRGW instances on the virtualization node and 30 matching linux containers simulating subscribers.

   Since a VLAN trunk is being used, it's important not to forget about the 4 bytes overhead in packets thus the redefinition of MTU size.

2. Build the OpenWrt base image and linked clones.

Qemu/KVM base images

Base imaging was inherited from the qemu userspace component as an added feature of KVM. The idea behind the base image is simple. A disk image is built as a normal image would be built. Then a new image is built using the first as base image, also known as the backing file. From this point on, KVM will only read from the base image (backing file). If anything needs to be written to an image, KVM will only write to the new image. It is, in effect, similar to a diff where all new data is written only to the new image while preserving the state of the backing file. As a side effect, the working image will be the same size as the base image. This is where the qemu copy on write term came from.

Building OpenWrt base image

OpenWrt base image should be based on a template disk image that is a practical starting point for current needs. It's always good to start with a base image that is at a state right after a fresh install; no patches, no applications, just the state of the machine right after installation of the OS. To be able to create the base image for OpenWrt *qemu-img* command would be used as bellow:

```
qemu-img create -f qcow2 openwrt-master.qcow2 100M
```

This image will be the base image for OpenWrt. Once the image is created, it is booted and OpenWrt Backfire 10.03 x86 image[23] is dumped into it.

Create a new image from the template Now that the base image is ready, a new one can be created starting from the template. The new image would be created with the following command using the -b flag.

```
qemu-img create -b openwrt-master.qcow2 -f  qcow2   openwrt-clone.qcow2
```

This will create a new copy-on-write image called openwrt-clone.qcow2 which uses the original openwrt-master as a template base image. It's not necessary to specify a filesize with this command as it will automatically set the new image to the same size as the base image. At this point the following command could be run on the new image to note the reported size.

```
# qemu-img info openwrt-clone.qcow2

image: openwrt-clone.qcow2
file format: qcow2
virtual size: 100M (104857600 bytes)
disk size: 56K
cluster_size: 4096
backing file: openwrt-master.qcow2 (actual path: openwrt-master.qcow2)
```

Note the size of the new image, 100M, the same size as the original base image and note how small the image is, 56K. At this point, the new image is basically a clone from the base image and dependent on the base image being in the same

state it was at the time the clone was created. For this reason, it's better never to try to boot the base image again as some data may be inadvertently written to disk causing some inconsistencies.

3. Use virt-manager to create a base template for the OpenWrt master VM

   At this point in time there's only a storage unit with OpenWrt 10.03 on it, the next step is to create a VM that uses that storage unit.
   Basically this completes number 2., *virt-manager* is a front-end manager for libvirt VMs written in python, one of the ways to create a new VM in libvirt is by using *virt-manager*. Using this tool, OpenWrt VM settings are easily setup and behind the scenes an XML template keeping all the defined settings is generated by the creation process.
   Finally, during the creation process of the new master VM in *virt-manager*, the base image created by the process described before with *qemu-img* command is configured to be used by the new master VM.

4. Setup the network cards to be used by the master VM.

   OpenWrt is not compliant with all the available NIC drivers in libvirt. To avoid problems i1000 NICs should be used. Every OpenWrt VM is setup with two NICs, one is a bridge to the LAN side and the other is a bridge to the WAN side, *eth0* and *eth1* respectively.
   It is also at this point that the bridge where the VLAN interface is on, is specified.

   The master OpenWrt VM is now ready to start.

5. Clone the master OpenWrt libvirt image.

   In order to have any number of OpenWrt VMs managed by libvirt, another bash script has been created to make the procedure more agile. Using this script the 30 OpenWrt VMs used should be now created. To wrap everything up, each new cloned VM will also be using an OpenWrt image clone created following the process described in number 2..
   Listing B.2 shows the script.

Listing B.2: bash script to make any number of libvirt clones.

```
#!/bin/bash

MASTER_IMAGE_FILENAME="openwrt-backfire_master"
CLONE_IMAGE_FILENAME="openwrt-backfire_10.03_x86_instance"

for i in $(seq 1 1 30)
do
        if [ ${#i} -eq 1 ]
        then
                virt-clone -o $MASTER_IMAGE_FILENAME -n \
                "${CLONE_IMAGE_FILENAME}_${i}" -f \
                /var/lib/libvirt/images/"${CLONE_IMAGE_FILENAME}_${i}.qcow2" \
                --preserve-data --connect=qemu:///system
        else
                virt-clone -o $MASTER_IMAGE_FILENAME -n \
                "${CLONE_IMAGE_FILENAME}_0${i}" -f \
                /var/lib/libvirt/images/"${CLONE_IMAGE_FILENAME}_${i}.qcow2" \
                --preserve-data --connect=qemu:///system
                echo "nothing to do $i"
        fi
done
```

After this process it's now time to rename the bridge name inside each VM XML template because the clone process keeps the same network interfaces used by the master VM and each clone must be attached to a unique bridge, created by the process described in number 1..
Again a bash script was used. Listing B.3 shows the script.

Listing B.3: bash script to rename each bridge on a libvirt VM template.

```
#!/bin/bash

PATH_LIBVIRT_CFG="/etc/libvirt/qemu"
CLONE_IMAGE_FILENAME="openwrt-backfire_10.03_x86_instance"

for i in $(seq 1 1 30)
do
        if [ ${#i} -eq 1 ]
        then
                find "${PATH_LIBVIRT_CFG}/${CLONE_IMAGE_FILENAME}_0${i}.xml" \
                -type f | xargs sed -i "s/br0/br${i}/g"
        else
                find "${PATH_LIBVIRT_CFG}/${CLONE_IMAGE_FILENAME}_${i}.xml" \
                -type f | xargs sed -i "s/br0/br${i}/g"
        fi
done
```

After applying this receipt, the Linux workstation VMs will now be able to get an ip address given by each DHCP server on the OpenWrt VM instances and access dei's network and the internet.

6. Start all OpenWrt VM instances

At this point the desired number of OpenWrt VMs will be ready to start. Another simple bash script has been created for the purpose.
Listing B.4 shows the script.

Listing B.4: bash script to start any number of libvirt clones.

```
#!/bin/bash

IMAGE_CLONE_NAME="openwrt-backfire_10.03_x86_instance"

for i in $(seq 1 1 10)
do
        if [ ${#i} -eq 1 ]
        then
                virsh start "${IMAGE_CLONE_NAME}_0${i}"
        else
                virsh start "${IMAGE_CLONE_NAME}_${i}"
                echo "nothing to do $i"
        fi
done
```

After this step, all the vRGWs will be running a ready to accept subscribers.

7. Install OpenVZ on the second Linux Server.

   At this point, it is time to setup the second server with a LXC framework to simulate subscribers for the previously created vRGWs. OpenVZ should be installed with the *centos-6-x86_64* OS template available for download on OpenVZ website.

8. Create Linux Containers to simulate subscribers.

   The next step is to create linux containers on the second server from Figure 4.1.1, to simulate a different subscriber for each vRGW instance. Therefore using OpenVZ as an LXC factory we can take advantage from the next bash script, B.5, to achieve just that:

Listing B.5: bash script to build OpenVZ linux containers.

```
#!/bin/bash

for i in $(seq 1 1 30)
do
        if [ ${#i} -eq 1 ]
        then
                vzctl create 10${i} --ostemplate centos-6-x86_64 \
                --hostname vz10${i}
                vzctl set 10${i} --netif_add eth0 --save
                vzctl start 10${i}
                sleep 5
                brctl addif br${i} veth10${i}.0
                vzctl exec 10${i} ifconfig eth0 mtu 1496
                vzctl exec 10${i} dhclient eth0
        else
                vzctl create 1${i} --ostemplate centos-6-x86_64 \
                --hostname vz1${i}
                vzctl set 1${i} --netif_add eth0 --save
                vzctl start 1${i}
                sleep 5
                brctl addif br${i} veth1${i}.0
                vzctl exec 1${i} ifconfig eth0 mtu 1496
                vzctl exec 1${i} dhclient eth0
        fi
done
```

The script from Listing B.5 also binds each container to one VLAN coming from the VLAN trunk between the virtualization node and the OpenVZ Server. After the VLAN bind *dhclient* is called on each container to request an IP address on the correspondent vRGW on the virtualization node.

9. Simulate traffic on the subscriber side.

   At this point *iperf* is ready to be used on each linux container to generate traffic between the container and the two iperf Linux boxes, acting as traffic sinks, passing through each correspondent vRGW on the virtualization node.

References

[1] G. Young. *Broadband Forum Overview with Focus on Next Genera-tion Access.* http://www.uknof.org.uk/uknof14/Young-BroadbandForum.pdf. September 2009. ix, 17, 18

[2] D. Abgrall. *Virtual Home Gateway: How can Home Gateway virtualization be achieved?, Deliverable D1 of EURESCOM study P2055* (September 2011). 2, 13

[3] T. Cruz et.al. *CWMP Extensions for Enhanced Management of Domestic Network Services, LCN 2010* (2010). 10, 14

[4] *OpenStack project.* http://www.openstack.org. 11, 27

[5] *oVirt project.* http://www.ovirt.org. 11, 27

[6] *The Home Gateway Initiative (HGI).* http://www.homegatewayinitiative.org. 13

[7] P. Pastorino. *The Role of OSGi Technology in the Home Gateway Initiative (HGI) and End to End Connectivity and Service Provisioning, OSGI Alliance* (2005). 13

[8] T. Cruz et.al. *How to Provision and Manage Off-the-Shelf SIP Phones in Domestic and SOHO Environments, LCN 2010* (2010). 14

[9] Broadband Forum. *Applying TR-069 to Remote Management of Home Networking Devices (TR-111)* (December 2005). 14

[10] Broadband Forum. *TR-069 - CPE WAN Management Protocol specification v1.3, Amendment 4* (July 2011). 14, 23, 47

[11] *Broadband Forum.* http://www.broadband-forum.org. 16

[12] Broadband Forum. *Migration to Ethernet-Based Broadband Aggregation, TR-101 Issue 2* (July 2011). 16

[13] Broadband Forum. *Using GPON Access in the context of TR-101, TR-156* (September 2010). 16

[14] IEEE 802.1 Working Group. *IEEE Std. 802.1Q-2011, Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks* (August 2011). 18

[15] IEEE 802.1 Working Group. *IEEE Std. 802.1ad-2005, Virtual Bridged Local Area Networks Amendment 4: Provider Bridges* (March 2006). 18

[16] Joseph, Vinod. *Deploying Next Generation Multicast-Enabled Applications: Label Switched Multicast for MPLS VPNs, VPLS, and Wholesale Ethernet* (Elsevier, Inc., 2011). 19

[17] *CentOS project*. http://www.centos.org. 26

[18] A. Kivity. *KVM, One Year On, Presented at KVM Forum 2007, Tucson, USA* (August 2007). 26

[19] *Libvirt project*. http://libvirt.org. 26

[20] K. Kolyshin. *Virtualization in Linux*. http://download.openvz.org/doc/openvz-intro.pdf (September 2006). 26

[21] *Iperf*. http://iperf.sourceforge.net. 27

[22] ITU. *Quality of Experience Requirements for IPTV Services, FG IPTV-DOC-0184* (December 2007). 27

[23] *OpenWrt project*. https://openwrt.org. 27, 44, 69

[24] *Virt-top*. http://people.redhat.com/rjones/virt-top. 27

[25] *Open Nebula project*. http://opennebula.org. 27

[26] M. Ellis, D. Pezaros, C. Perkins. *Performance Analysis of AL-FEC for RTP-based Streaming Video Traffic to Residential Users, in Proc. of the 19th Int. Packet Video Workshop (PV), Munich, Germany* (2012). 29

[27] B. Nagel. *Demonstration of TVoIP services in a multimedia broadband enabled access network, MUSE Project Presentation at Broadband Europe 2007, Antwerp* (December 2007). 29

[28] *JBoss Application Server*. http://www.jboss.org. 31

[29] *Glassfish Application Server*. http://glassfish.java.net. 31

[30] *Apache XML-RPC Java implementation*. http://ws.apache.org/xmlrpc. 31

[31] *KVM project*. http://www.linux-kvm.org. 31

[32] *Unix top project*. http://www.unixtop.org. 31, 36

[33] *Tcpdump project*. http://www.tcpdump.org. 31

[34] *Python Programming Language*. http://www.python.org. 33

[35] *dd-wrt project*. http://www.dd-wrt.com. 44