Managed Hybrid Storage for Home and SOHO Environments

Tiago Cruz, Paulo Simões,
João Rodrigues, Edmundo Monteiro
DEI-CISUC – University of Coimbra
Coimbra, Portugal

Fernando Bastos
Alexandre Laranjeira
PT Inovação
Aveiro, Portugal

***Abstract*—From specific-purpose storage appliances to the new generation of cloud storage services, the industry has different solutions for the data storage needs of home and small office users, whereas for backup, resource consolidation or file sharing purposes. Although dedicated appliances and cloud services offer a different balance between capacity, accessibility, and cost, the two paradigms are complementary, rather than incompatible.**

This paper proposes an operator-managed hybrid storage service which explores the existing complementarity between the appliance-based and the cloud-based storage service models. By making use of home gateways/routers as storage service appliances, it aims to provide an integrated solution eliminating the need for dedicated storage devices. The proposed approach fully integrates with the prevailing management frameworks already used by broadband access network operators, namely the CPE Wan Management Protocol (CWMP).

***Keywords*— Network Storage, CWMP, Home Networks**

I. Introduction

Right from the start, networked storage was one of the main driving forces behind the success of the small workgroup and home Local Area Networks (LANs). What started as basic file sharing between PCs has evolved towards dedicated devices such as home servers or storage appliances, providing consolidated storage services with a variable degree of redundancy and data recovery capabilities.

In addition, a cloud-based storage paradigm has recently emerged, exemplified by services such as Dropbox [1], Amazon's Cloud Drive [2], Apple's iCloud [3] or Ubuntu One [4], which provide users with a new type of storage service, allowing access from almost any place with an Internet connection (and not just inside the user premises, as it happens with the majority of dedicated storage appliances) to virtualized storage containers in the cloud.

For home and Small Office, Home Office (SOHO) users, both paradigms (dedicated appliances, cloud services) have their share of advantages and drawbacks in terms of accessibility, capacity, cost and reliability. Some of them relate to its usage scope – for instance, inside the LAN, cloud storage services are no match for dedicated appliances in terms of available space and access speeds, while outside the customer premises they have the upper hand in terms of accessibility. Despite their differences, the two paradigms are complementary, hinting at the possibility of creating an optimized solution by bringing them together.

This paper presents an operator-managed hybrid storage solution that offers a balance between cost, reliability and accessibility by eliminating the need for a dedicated appliance inside the customer premises and transforming the home router (or residential gateway, RGW) in a storage appliance which, similarly to conventional NAS devices (like Apple's Time Capsule [5]), has its own local storage capabilities, but unlike them, is able to synchronize with a virtualized storage container on the operator storage service infrastructure.

The advantages are manifold: first, it provides redundancy and reliability by replicating data to a virtual container located outside the customer premises, on the storage provider infrastructure; second, it relieves users from the task of configuring and managing their own storage devices; and third it takes advantage of the fact that the RGW is a device which is permanently powered on (especially in triple-play environments) in order to provide connectivity services for the LAN and eliminate the need for a separate appliance.

Remote service, device management, configuration and provisioning mechanisms for operators are provided by means of Broadband Forum's [6] CPE Wan Management Protocol (CWMP [7]), the *de facto* standard for management of devices located in the customer premises on broadband access network environments. This enables operators to use its existing infrastructure to provide a complete management solution.

The rest of this paper is organized as follows. Section 2 discusses network storage paradigms and presents the proposed hybrid storage solution. Section 3 addresses CWMP-based management and eventing mechanisms. Section 4 presents potential application scenarios, while Section 5 discusses implementation and validation. Section 6 addresses related work and Section 7 concludes the paper.

II. A Managed Hybrid Storage Framework

This section discusses the main home/SOHO networked storage paradigms, providing the basic reasoning for the proposed framework, whose operation is also described.

A. Storage paradigms in home/SOHO environments

Storage, even in the form of simple PC-to-PC file sharing, was one of the most important services behind the emergence of home LANs, having evolved up to the present scenario, with two dominant networked storage paradigms for home/SOHO users: dedicated devices and cloud services.

Specialized networked storage devices, such as home servers or dedicated appliances provide consolidated storage, frequently combined with features like redundancy, file versioning or media streaming. Inside the LAN, these storage appliances have the edge in terms of the capacity/speed ratio, providing fast access (since they are unconstrained by access network speeds) to bigger amounts of storage capacities. Still, for locations other than the customer LAN, accessibility is trade-off that most users are unable to deal with, partly due to the use of network protocols such as CIFS/SMB [8], typically inadequate for use outside LANs.

On the other hand, cloud storage services address the accessibility gap, by providing a virtualized container where users can store their files and which is conveniently accessible from anywhere with an Internet connection, using a dedicated client application or a browser interface. Also, cloud storage services make use of location-independent distributed redundancy mechanisms to provide better data protection than storage appliances, which are more vulnerable to catastrophic events and/or human errors. However, cloud storage is susceptible to quality of service issues, since it is provided as an over-the-top, best effort service that is not directly supported by operators, thus being unable to benefit from end-to-end quality of service mechanisms.

This situation hints at the potential benefits of a balanced, hybrid solution integrating the best characteristics of each storage paradigm. This also prefigures an opportunity for operators to get involved, by providing a managed storage service for customers, with increased value against over-the-top solutions, as discussed on the next section.

B. *Residential gateways as storage hubs*

In order to achieve an optimal balance between the cloud storage and dedicated appliance paradigms, we propose a solution that transforms the RGW in a hybrid storage appliance. The motivation for such an approach is manifold. First, modern RGWs are embedded systems whose hardware has evolved to the point where its computing (CPU, memory) and expansibility (e.g. USB ports) capabilities are up to the requirements of such a solution. Second, they are located between the access network and the customer premises LAN domains, thus being ideal management entry points for service deployment and support. Finally, RGWs already operate on a continuous basis, foregoing additional dedicated devices.

In this operation model, each RGW has its own local storage capabilities (embedded or attached) in the form of flash memory or hard disk. Its content is made available to the customer premises LAN, using protocols such as CIFS/SMB, FTP [9] or HTTP(s) [10] and synchronized with the cloud storage container on the operator infrastructure (see Figure 1).

Every RGW associated with the same storage service subscription is allowed to synchronize with the main repository. Roaming users might also use native client applications to access their data while on the road (however, this use case is outside the scope of this paper).

Each RGW is able to detect changes on its local storage space and start a synchronism operation with the main service repository, which will be synced with the most recent version. Subsequently, the service will trigger a global sync operation with all RGWs that are registered to the same user, to update their local storage from the main storage service container. While the operator is in charge of the storage service frontend and its management, the backend might be single-sourced by the operator or provided by a 3rd party storage provider, like Amazon's S3 service [11].

This solution has the ability to stream content to the customer premises network at the same speed than a dedicated NAS device would do (once the content is replicated on local storage). It supports disconnected operation, remaining accessible in the event of an access network failure. Also, it does not require LAN clients to have specific support for the service, since data is made accessible using standard protocols (such as CIFS/SMB). Redundancy is provided in a cost-effective way, since data is replicated to the service main storage repository, instead of relying only on self-contained device data replication methods such as RAID (which, nevertheless, can be used together with this solution to achieve extra redundancy). This makes this solution adequate for both multi-branch SOHO infrastructures and home users with a service subscription for a single household. Versioning support allows the retrieval of earlier versions of specific files.

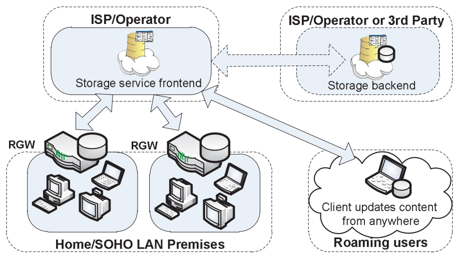


Figure 1: Hybrid storage service operation model

III. MANAGEMENT AND EVENTING

This section deals with management and eventing mechanisms, which are both integrated within the CWMP protocol framework. Starting with a brief introduction to the CWMP protocol, it will next delve into the design of a CWMP extension for management of service storage endpoints, analysing its architecture, data model integration and operation. Eventing mechanisms are also discussed.

A. *The CWMP protocol*

Broadband Forums' CWMP protocol suite is the established standard for secure device and service management on broadband environments. Its adoption by the storage service hereby proposed enables operators to use their already existing Operations Support Systems (OSS) to handle service management and storage synchronization events.

CWMP provides a management API of Remote Procedure Calls (RPCs) supported by a set of data models defined by related standards such as TR-106 [12] or TR-157 [13]. Management sessions are always initiated by the managed device, either directly or by request of the management server (designated by Auto-Configuration Server, ACS). The CWMP framework already encompasses a standardized data model for management of Network Attached Storage (NAS) devices, specified by TR-140 [14] – even though very few commercial NAS devices actually support CWMP up to now.

TR-140 is the referential data model for the storage service hereby proposed. Secure eventing interfaces for synchronism between local storage spaces and the operator are also provided by the CWMP protocol, using a native mechanism which allows for the definition of notification attributes on parameters, either by default or explicitly, using the *SetParameterAttributes* CWMP method. For instance, when the state of a given parameter with the *Active Notification* attribute enabled changes, the CWMP device will schedule the execution of an *Inform* method on the ACS to notify it. This mechanism is used to generate synchronization events for subscriber devices involved in the storage service.

B. Integration architecture

The proposed architecture integrates together a CWMP interface with the storage service frontend. At the RGW level, there is a specific integration component that acts as module of the gateway CWMP agent, interacting with the operator ACS and with the local storage service components, so that they can be remotely managed. This specialized component is responsible for its own data model extensions (according to TR-106 and TR-140) and configuring the RGW LAN storage server components accordingly (see Figure 2). On the operator side, the ACS interfaces with the storage service frontend using a middleware layer based on a message-oriented queuing system for passing events.

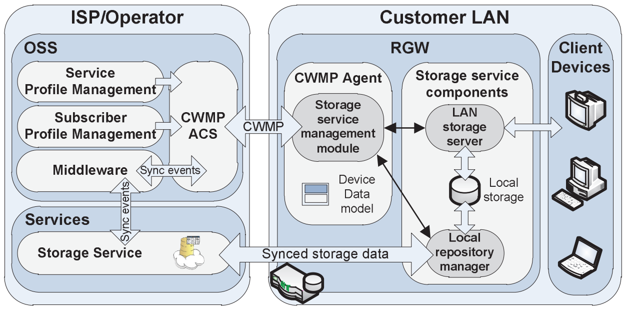


Figure 2: Integration architecture for the hybrid storage service

C. CWMP data model integration

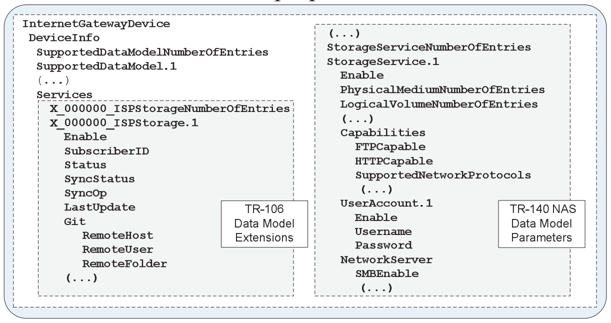
Considering the CWMP Data Model objects and parameters, the integration module will be responsible for two different branches of the supported CPE (RGW) data model: the standard branch for NAS devices (as defined by TR-140) and another set of TR-106 compliant extensions for service provisioning and management purposes (see Figure 3).

Both TR-140 and TR-106 explicitly allow for embedding the information of managed services within the TR-098 data model of RGWs (even though, to the best of our knowledge, our proposal is the first to use this possibility for supporting distributed storage features), under the *Services* root object.

The data model extensions are organized using the *X_000000_ISPStorage* object on the CWMP data model (with an Organizational Unique Identifier (OUI) [15] defined as 000000 for test purposes). These data model extensions are also declared in a *SupportedDataModel* table entry (of the *DeviceInfo* root object), which points to an URL containing their XML description hosted on the CPE itself, so that the ACS can gain knowledge about the device data model supported by the managed device. The same happens for the TR-140 standard data model, whose CPE support has to be declared before being instantiated.

The information needed to control the storage service components are embedded on the CWMP data model of the RGW using an *X_000000_ISPStorage* entry, while specific local scope storage service and device configuration parameters are managed using TR-140 data model structures.

The TR-140 *StorageService* entry is used to manage the LAN storage service which will provide access to the repository data from the customer premises LAN. Therefore, the *Baseline:1, NetServer:1, FTPServer:1, HTTPServer:1, UserAccess:1, VolumeConfig:1* and *VolumeThresh:1* TR-140 profiles are implemented to provide support in the data model for the LAN network server properties.



```
InternetGatewayDevice
 DeviceInfo
  SupportedDataModelNumberOfEntries
  SupportedDataModel.1
  (...)
  Services
    X_000000_ISPStorageNumberOfEntries
    X_000000_ISPStorage.1
      Enable
      SubscriberID
      Status
      SyncStatus
      SyncOp
      LastUpdate
      Git
         RemoteHost
         RemoteUser
         RemoteFolder
      (...)

(...)
StorageServiceNumberOfEntries
StorageService.1
  Enable
  PhysicalMediumNumberOfEntries
  LogicalVolumeNumberOfEntries
  (...)
  Capabilities
    FTPCapable
    HTTPCapable
    SupportedNetworkProtocols
     (...)
  UserAccount.1
    Enable
    Username
    Password
  NetworkServer
    SMBEnable
     (...)
```

TR-106 Data Model Extensions
TR-140 NAS Data Model Parameters

Figure 3: CWMP data model integration

D. Service Management using CWMP

Service management can be split in three different phases: provisioning, service initialization, and runtime management.

The **provisioning process** goes as follows (see Figure 4): once the CWMP storage service management extension is active, the *X_000000_ISPStorage* and the *StorageService* objects are instantiated and enabled on the CWMP data model of the RGW, for management of the storage service and the network storage server component, respectively. Then, the ACS configures the service, which is then activated by setting the *X_000000_ISPStorage.1.Enabled* parameter to *True*.

The **initialization process** goes as follows (see Figure 4): on each initialization, the CWMP storage service management module configures the repository manager and the network storage server using the information from the CWMP data model instances. Once the internal modules are successfully initialized, the CWMP storage management module changes the *X_000000_ISPStorage.1.Status* parameter (which has the Active Notification attribute enabled) to *Ready*. When the state of a given parameter with the Active Notification attribute enabled changes, the CWMP device notifies the ACS by executing an *Inform* method. This happens with the *Status* parameter once its value changes to *Ready* as the result of the initialization process. On its turn, the Middleware layer passes the information to the Storage Service Frontend, assembling a subscription request on behalf of the RGW. If the RGW is allowed to access the service, a confirmation message is issued and passed to the ACS, changing the *Status* parameter to the *Subscribed* state. After each initialization, the repository manager attempts to synchronize its data with the storage service, in order to update its content to the latest version.

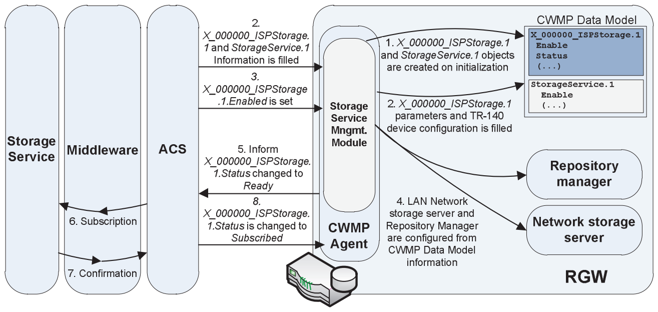

Storage Service
Middleware
ACS
2. X_000000_ISPStorage.1 and StorageService.1 Information is filled
3. X_000000_ISPStorage.1.Enabled is set
5. Inform X_000000_ISPStorage.1.Status changed to Ready
8. X_000000_ISPStorage.1.Status is changed to Subscribed
6. Subscription
7. Confirmation
Storage Service Mngmt. Module
CWMP Agent
1. X_000000_ISPStorage.1 and StorageService.1 objects are created on initialization
2. X_000000_ISPStorage.1 parameters and TR-140 device configuration is filled
4. LAN Network storage server and Repository Manager are configured from CWMP Data Model information
CWMP Data Model
X_000000_ISPStorage.1
Enable
Status
(...)
StorageService.1
Enable
(...)
Repository manager
Network storage server
RGW

Figure 4: Service management via CWMP

Runtime management goes as follows (see Figure 4): for setting specific parameters on the CWMP data model objects for the storage service (*X_000000_ISPStorage.1* and *StorageService.1*), the ACS interacts with the CWMP agent on the domestic gateway using standard CWMP methods. For each ACS-instructed configuration change, the CWMP Storage Service management module deals with subsequent internal module configuration modifications.

E. Eventing using CWMP

While the eventing mechanism for signalling synchronization operations between local storage repositories and the operator service could be implemented using a custom signalling method, the fact is that CWMP proved to be adequate for the purpose, since it already offers a secure event notification mechanism. As such, the middleware provides operation translation and queuing between the storage service frontend and the ACS, similarly to a decoupling layer that can be implemented in a distributed fashion using a publish-subscribe message queuing system such as ActiveMQ [16].

The synchronization mechanism for RGWs is based in a push-pull model. However, subscribers with a single storage service access point (like regular home users without roaming access) may generate push operations almost exclusively.

Push operations (see Figure 5) occur when the content of the local repository is changed.

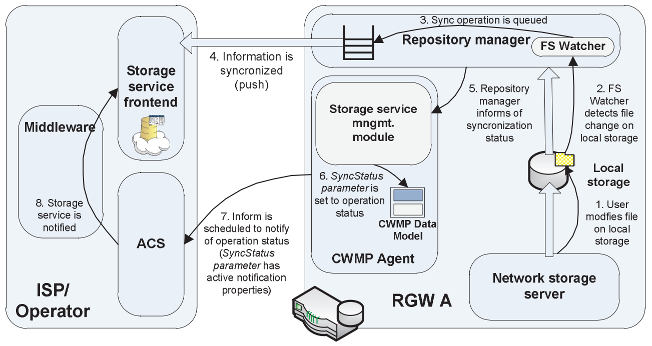

3. Sync operation is queued
Repository manager
FS Watcher
4. Information is syncronized (push)
Storage service frontend
Middleware
5. Repository manager informs of syncronization status
2. FS Watcher detects file change on local storage
Storage service mngmt. module
Local storage
6. SyncStatus parameter is set to operation status
CWMP Data Model
1. User modfies file on local storage
8. Storage service is notified
ACS
7. Inform is scheduled to notify of operation status (SyncStatus parameter has active notification properties)
CWMP Agent
Network storage server
ISP/ Operator
RGW A

Figure 5: Push event notification via CWMP

The user accesses the local storage device by means of the network storage server (for instance, a SMB/CIFS server), being able to modify its contents. A filesystem watcher on the repository manager is able to detect changes and, as soon a file handle is closed, an update operation is pushed into the internal stack (a FIFO).

Every time a content change is detected, it is pushed into the storage service frontend to update its version. Once the operation is finished, the CWMP storage service management module is notified, in order to update the contents of the *SyncStatus* parameter (see Figure 3), which has Active Notification properties enabled – forcing the CWMP agent to notify the ACS of its change. The middleware then passes this information to the storage service frontend to inform of the sync operation status – this mechanism enables transactional properties, also helping manage concurrency issues.

The storage service frontend keeps track of all subscribers. When there is more than one service subscriber entity, a push operation must be issued to all involved subscribers, to update their local information. **Pull operations** (see Figure 6) are issued by the storage service frontend and queued by the middleware layer, generating an internal operation sequence number (*opsec*), appended to each operation. The ACS is instructed to execute a *SetParameterValues* operation on the *SyncOp* parameter, which is filled with the string *pull <opseq>*. As a result, the CWMP storage service management module of the RGW queues a pull operation on the local repository manager.

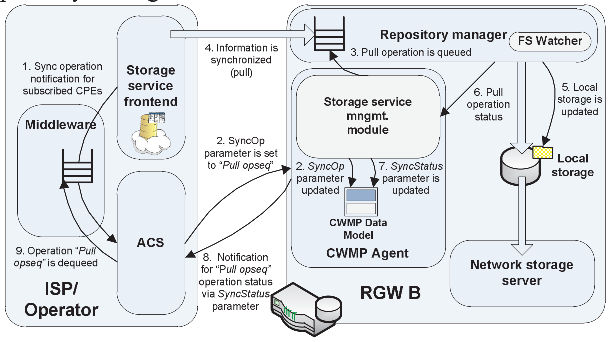

Repository manager
FS Watcher
4. Information is synchronized (pull)
3. Pull operation is queued
1. Sync operation notification for subscribed CPEs
Storage service frontend
Middleware
6. Pull operation status
5. Local storage is updated
Storage service mngmt. module
2. SyncOp parameter is set to "Pull opseq"
2. SyncOp parameter updated
7. SyncStatus parameter is updated
Local storage
CWMP Data Model
ACS
CWMP Agent
9. Operation "Pull opseq" is dequeed
8. Notification for "Pull opseq" operation status via SyncStatus parameter
Network storage server
ISP/ Operator
RGW B

Figure 6: Pull event notification via CWMP

Once the pull operation is completed the repository manager makes the updated information available on the local storage device and communicates the operation status to the CWMP storage service management module, which updates the *SyncStatus* parameter with the operation result, appended by its *opseq* sequence tag. Since this parameter has Active Notification properties, the CWMP agent notifies the ACS of its change, which will pass the information into the middleware layer, therefore dequeuing the operation.

Local conflicts are resolved by each subscriber agent entity, which has the ability to detect if a file on a local instance was updated (or locked) during a synchronization operation involving the same file. In this case, a collision is registered and the offending file is renamed to its original name, concatenated with a timestamp and queued for future update on the main storage container.

IV. Application scenarios

This section discusses how operators can use the proposed solution to enrich their service portfolio for home and SOHO users. Specifically, three scenarios are presented.

A. Enterprise storage synchronization for multiple branches

An organization with several dispersed branches could benefit from the hybrid storage service hereby described as a means to keep a shared document repository synchronized (and safe) between all workgroup networks. This service could also be used to distribute software updates, making them available at local speeds, once the related files are propagated.

An operator could market this service to small and medium enterprises in a communication service bundle. Since the access provider is the same entity as the storage service provider, it can provide a complete managed solution with end-to-end QoS enforcement. This use case may also apply for home subscribers, enabling for instance, to replicate a media library between a permanent and a holiday residence.

B. Hybrid NAS for home users

Most low-cost dedicated NAS appliances are not redundant by nature, being little more than network-attached disks with a special firmware component. The hybrid storage model hereby presented can be used to add redundancy, replicating data to an external storage container. Since RGWs could provide this functionality, an operator could market this service in a converged service bundle.

However, even dedicated appliances with RAID support may benefit of the storage model hereby proposed, adding an extra redundancy layer (like a cold site) to its native mechanisms, to protect data from catastrophic events. While the proposed framework is targeted towards RGWs, it can be equally implemented in dedicated NAS devices.

C. Content distribution mechanism for home users

This storage service could be used by an operator as a content delivery mechanism for media distribution, such as Video on Demand. Users could, for instance, buy movies from an online store and have it placed on its storage container, being propagated to all devices associated with the subscriber account. Also, operators could use this service to improve support, automatically providing software and firmware updates to devices that the user might have acquired right to the customer premises, in a transparent and convenient way.

V. Validation

This section discusses the prototype implementation of the hybrid storage service presented on this paper, the experimental validation process and obtained results.

A. Proof-of-concept implementation

A proof-of-concept implementation of the proposed architecture (Figure 2) was designed to provide a test ground.

The local repository management module uses the *git* [17] version control system for data synchronization, with some customizations (like disabled merge capabilities). *Git* uses bandwidth and CPU resources efficiently, only transferring file changes (compressed binary deltas) on each update, with encryption provided by an SSH [18] tunnel. Since *git* was designed for revision control, it provides file versioning mechanisms, offering all the necessary means to deal with update conflicts – when *git* is not able to do it by itself, it gives enough means to implement an external conflict detection and resolution component on the RGW. A filesystem watcher uses the *inotify* Linux kernel call [19] to detect changes on the synchronized volume and initiate update operations.

The LAN storage server module was implemented using available software components to provide HTTP (using *thttpd* [20]), FTP (using *vsftpd* [21]) and SMB/CIFS (using *Samba* [22]) support for LAN-side client access.

The proof-of-concept CWMP agent embedding the storage management module was developed in Java, using an in-house CWMP agent extensibility framework [23]. Still, it could also be implemented using any other CWMP agent library.

B. Testbed and test plan

Figure 7 presents the established testbed, that emulates the conditions of commercial broadband access networks based on Gigabit Passive Optical Networks (GPON) or xDSL technologies – including the operator (OSS, services) and customer premises domains. To mimic the conditions of access network links, a transparent *Dummynet* bridge [24] interconnects the customer with the ISP – this bridge proved to be capable of a sustained forwarding throughput of 800Mb/s. A PC with *Wireshark* [25] captures and measures network traffic, using a mirroring port on the Ethernet switch. A Linux system hosts the CWMP ACS and the related profile management services.

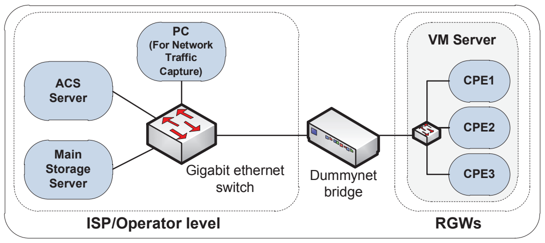


Figure 7: Implemented testbed

As for the CPEs (RGWs), it was very important to have identical hardware platforms for each one, in order not to contaminate results. This was by means of virtualization, emulating each RGW inside a virtual machine hosted by a quad-core Intel Core 2 Q8400 with 12GB of RAM and a PCIe Gigabit Ethernet interface. By using the VMware ESXi [26] hypervisor, each RGW was emulated as an exact replica of the others: thus, processor (with affinity of 1 core for each CPE, limited to 1500MHz), memory (512MB), storage (5GB for the system image, 10GB for the attached storage device) and hardware specifications were identical for all instances. The virtualized internal switch of the hypervisor provides equal network QoS access and throughput for each CPE [27, 28].

C. Tests and results

The test plan was divided in functional and performance tests. Functional tests validated the proposed operation model and integration schemes. Performance tests focused on overall operation and data overhead, with the following methodology:

- First, reference results were obtained on a scenario where all equipment were connected using 100Mb/s Ethernet (*Dummynet* bridge configured to limit traffic to 100Mb/s in both directions, for each CPE). In this test phase, results were obtained for local storage overhead, network traffic overhead and data synchronization performance.
- Next, tests were performed to measure data synchronization performance. The *Dummynet* bridge was configured to enforce bandwidth and traffic conditions representing typical commercial offers (see Table I). The rationale for the *Dummynet* configurations is discussed at [29].

TABLE I. BROADBAND TEST REFERENCE SCENARIOS

	Nominal bandwidth (b/s) (Down/Up)		Effective bandwidth (b/s) (Down/Up)		RTT Latency	Pkt. Loss
ADSL	8M	512K	6.68M	427.5K	20ms	0.1%
	16M	1M	13.36M	835K	20ms	0.1%
GPON	30M	3M	27.9M	2.79M	5ms	0%
	100M	10M	93M	9.3M	5ms	0%
LAN	100M	100M	100M	100M	<1ms	0%

All data synchronization tests were performed using a scenario with 3 CPEs, were one of them triggers an update which is propagated to the main repository and to the remainder CPEs, referred hereafter as the "1-to-2" scenario.

Local storage overhead was measured to assess how much space is used on the attached storage device on each CPE for a certain amount of used storage. The basic unit for storage comparison was based on decimal multiples of 100KB, both for total aggregate storage and individual file sizes. Tests were performed for two situations: **individual files**, to assess how compression influences the local storage overhead (since *git* uses compression on the local repository); and **multiple files**, to understand how the overhead differs for several files, for the same aggregate storage size. Both tests were performed on the reference 100Mb/s LAN scenario.

Figure 8 illustrates obtained results for single-file tests.

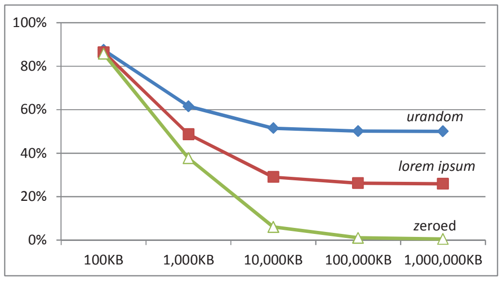


Figure 8: Overhead for individual files

Different sizes (all decimal multiples of 100KB) were used for three different content types: random content (difficult to compress and sourced from the */dev/urandom* Linux pseudo-device), random text (using a *lorem ipsum* text generator [30]), and zeroed files (best-case compression).

For small files, the higher relative overhead is due to extra information used by git to control versioning and integrity (which is negligible in absolute terms). For bigger files, local overhead depends on their content. Since *urandom* files have low compression ratios, local overhead is higher than for text files (50% against 26%, for best-case scenarios).

TABLE III. MULTIPLE FILE TEST SCENARIOS (WORST COMPRESSION RATE)

Aggregated size	Number of files (*urandom*)				
	1	10	100	1,000	10,000
1,000,000KB	1,000,000KB	100,000KB	10,000KB	1,000KB	100KB
100,000KB	100,000KB	10,000KB	1,000KB	100KB	
10,000KB	10,000KB	1,000KB	100KB		
1,000KB	1,000KB	100KB			
100KB	100KB				

For the multiple file tests the same *urandom*-based files were aggregated in sets with 1 to 10,000 files (see Table II) – see results in Figure 9. For the same amount of used space, local overhead difference for one and multiple files is negligible.

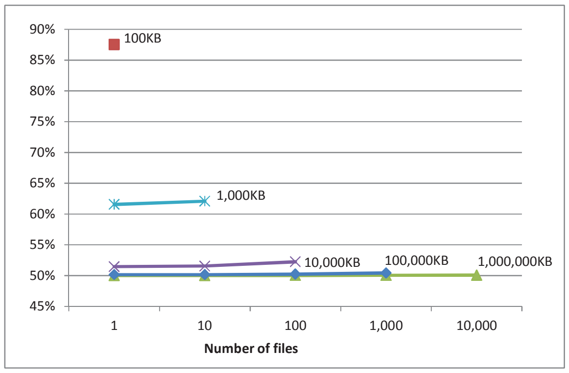


Figure 9: Multiple file overhead for the same aggregated size

Network traffic overhead was measured using the "1-to-2" test case on the reference 100Mb/s LAN scenario, comparing the total amount of synchronized data (aggregated upstream and downstream flows) with the total generated network traffic for the synchronization flows, for single and multiple files with random content (Figure 10).

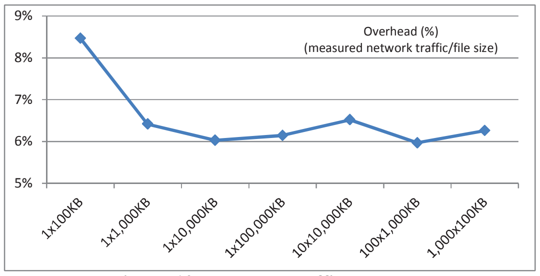


Figure 10: Network traffic overhead

Except for the case were a single 100KB file was transferred (where the overhead amounts to a mere 50KB, in absolute terms) the overhead remains consistently between 6% and 7%, regardless of the file size or the number of synchronized files.

CWMP protocol overhead was also measured, both for provisioning and synchronization scenarios. In both cases, 10 tests were performed: in the first case for a complete provisioning operation and, in the latter, for the "1-to-2" reference scenario. The average amount of CWMP-related traffic is very small in either situation, especially in comparison with data synchronization traffic. It accounts for 41KB (3.2% stdev) and 22KB (0.86% stdev) for provisioning and synchronization operations, respectively.

For data synchronization, we measured the complete delay for the “1-to-2” test case from the ACS standpoint and using CWMP events for timing. Tests were performed with multiple and single files with *urandom* content (with reduced compression ratio), with 5 interactions per each test.

In the **multiple file synchronization tests** (Figure 11) the distribution mechanism is not significantly affected by file sparsity, for the same aggregated amount of data (100000KB).

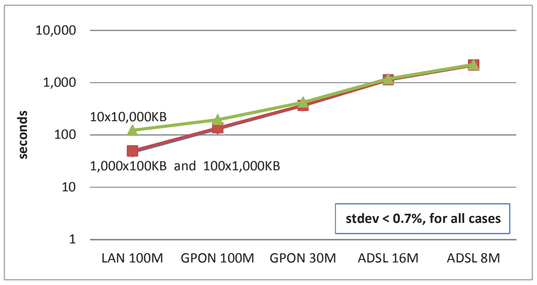


Figure 11: Synchronization results for multiple files (values in seconds)

A detailed analysis of synchronization performance, separated by download (*pull*) and upload (*push*) operations, revealed more details. For each case, the operation delay was compared with the theoretical network transfer time for the payload.

For *pull* operations the measurements are in general close to the theoretical network transfer time reference, with a small overhead for each network technology (Figure 12). The only exceptions are the LAN and faster GPON scenarios with the 10x10,000KB tests, due to processing limitations of the CPEs.

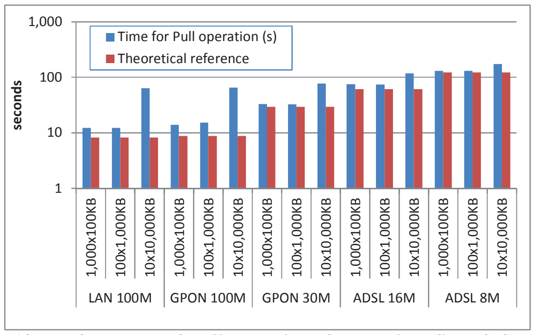


Figure 12: Performance of *pull* operations for multiple files (2 CPE avg.)

For *push* operations the overhead remains generally consistent, independently of the number of files on the payload (Figure 13).

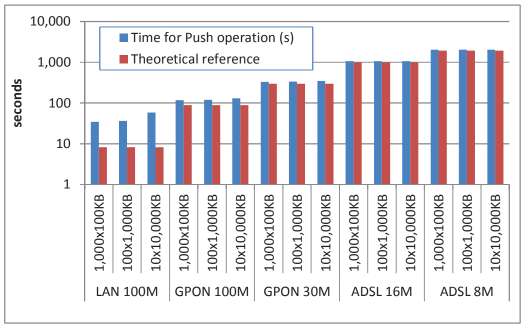


Figure 13: Performance of *push* operations for multiple files (2 CPE avg.)

However, for the reference LAN scenario there is an overhead increase – again due to CPE processing limitations. This behaviour was not observed on GPON scenarios because of asymmetric upload/download speeds.

Results for the **single file synchronization tests** (Figure 14) show that for file sizes above 100KB the synchronization time increases on the same order of magnitude as the file size. The 100KB exception is caused by the git transfer and processing overhead, more noticeable on small file sizes.

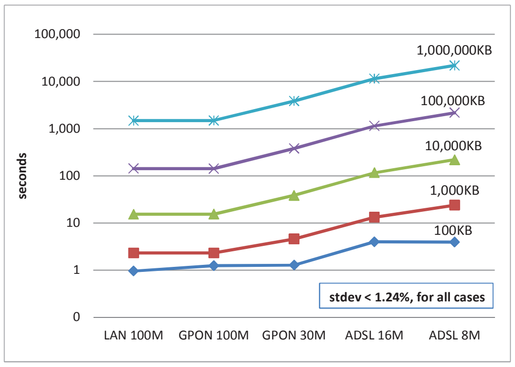


Figure 14: Single-file synchronization results (values in seconds)

A detailed analysis of single-file synchronization performance, for download (*pull*) and upload (*push*), revealed more details. For each case, operation delay was compared with the theoretical network transfer time for the payload.

Pull operations (Figure 15) suffer from the same problem observed with multiple files and faster networks, reflecting the processing limitations of the CPEs.

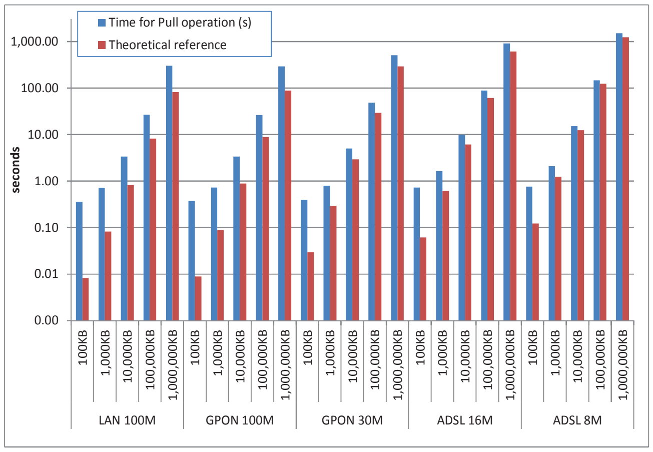


Figure 15: Performance of *pull* operations for single files (2 CPE avg.)

For *push* operations, only the LAN tests deviate significantly from the reference network transfer time (Figure 16). As with multiple files, this is due the asymmetrical upload/download speeds of GPON and ADSL.

Obtained results demonstrate that this approach has potential for real-world usage. The worst-case overhead for the proposed synchronization mechanism is within adequate levels and might be further mitigated in several ways: first, embedded system performance is increasing each day, eventually ceasing to be a limiting factor in faster networks; second, the synchronization mechanism is not expected to use all available bandwidth at all times (especially in triple-play scenarios), rather using the remaining capacity; third, since this is a fully managed solution, the operator is able to enforce QoS and other adequate measures to maximize performance.

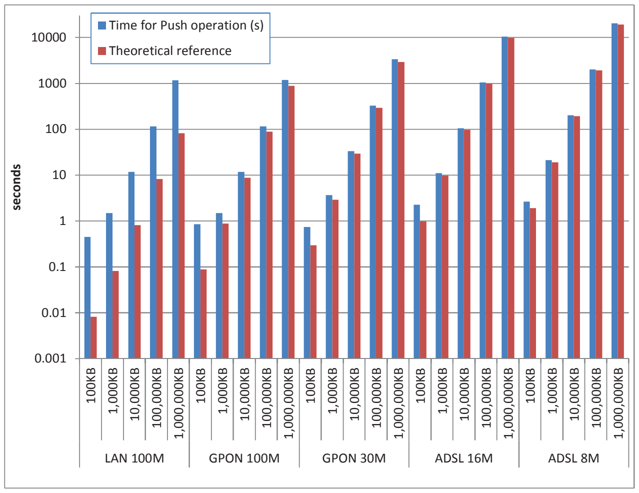


Figure 16: Performance of *push* operations for single files (2 CPE avg.)

VI. Related work

The research field of distributed storage systems has known significant activity in recent years. Support for disconnected access by using replication, one of the main novelties of CODA [31] (a descendent of AFS-2 [32], which already used disk caching) is now supported by other distributed filesystems, like Intermezzo [33]. Also, Bayou [34] and EnsemBlue [35] support total or partial peer-to-peer data propagation and replication in weakly connected scenarios. In general, these are either abandoned or unfit for production.

Clustered file systems have appeared in recent years, such as RedHat's GFS [36] (shared-disk), GlusterFS [37] or Lustre [38] (an evolution of Intermezzo). Also, the MapReduce [39] framework, spawned a new generation of filesystems such as Google's GFS [40] or Hadoop DFS [41]. Most of those approaches privilege specific features over others, such as access performance, fault-tolerance or parallelism. In general, existing distributed storage mechanisms with local replicas were not deemed adequate for our purposes, often because of reliance on close coupling of nodes, being designed for high-speed interconnection scenarios; others because replication schemes were incompatible with our purposes.

For the proposed solution there was no need for real time data synchronization, metadata servers or complex distributed lock control mechanisms, the problem being reduced to a question of file-level synchronization, for which *Unison* [42] and *Rsync* [43] were potential candidates. However, they lacked support for file versioning and replica reconciliation.

The proposed storage service relates with one of the use cases specified on TR-140: the remote backup storage service. However, it targets devices providing storage services for the LAN, without any explicit interfaces for service management or efficient synchronization mechanisms. Moreover, the *File Retrieval Theory of Operations* document section on TR-140 suggests the use of native CWMP download and upload methods to transfer files, with FTP and HTTP as an option for third parties. Both alternatives are inefficient for large quantities of data: file transfer over CWMP is affected by protocol and encoding overhead, and optimizations such as delta transfers or compression mechanisms are not supported.

To our knowledge, the use of storage gateways as participant nodes in a distributed storage topology is a novelty. While router-assisted distributed storage mechanisms were proposed [44], they did not consider the use of routers as storage nodes. Also, most work on the topic of distributed storage for home environments is focused on mechanisms for the LAN scope or mobile device accessibility [45], [35].

The commercial offer for cloud storage is mainly based on proprietary protocols and applications (even if access via browser is allowed, its not as convenient) targeting general-purpose client devices (like PCs, tablets or smartphones), without explicit support for storage appliances. This is the case with Dropbox [1], Ubuntu One [4] or Apple's iCloud [3]. The proposed solution goes further, supporting both the cloud and appliance-based paradigms, working in tandem.

Recently, some manufacturers [46] [47] started offering hybrid networked storage appliances capable of synchronizing their contents to external storage services – however, this feature is designed as a one way, over-the-top service with a single device in mind. Our proposal allows for two-way synchronization for multiple-devices, integrating its management and signalling mechanisms within the CWMP framework of the operator to provide a homogeneous solution.

VII. Conclusion

This paper proposed an operator-managed hybrid storage model, which makes use of RGWs as storage service hubs to explore the existing complementarity between the traditional/appliance-based and the cloud-storage service models. Also, it provides an integrated solution eliminating the need for special-purpose storage appliances by turning the RGW in a storage gateway that can be accessed by all kinds of devices inside the LAN without need for explicit support (like the installation of a client application).

The solution is fully managed by the operator, who is able to provision and manage the storage service, by integrating the proposed solution with the CWMP protocol suite, the prevailing standard for remote management of Customer Premises Devices in broadband access networks. Also, this enables the operator to provide complete end-to-end monitoring and QoS management.

Overall, feasibility of the proposed approach was demonstrated, in terms of local and network overhead and synchronization performance. Future work includes optimizations for *git* and an intelligent bandwidth management mechanism enabling the optimal use of available capacity.

Acknowledgements

This work was partially funded by PT Inovação (Project Virtuoso) and by Project QREN ICIS (Intelligent Computing in the Internet of Services – CENTRO-07-0224-FEDER-002003).

REFERENCES

[1] Dropbox Inc., http://www.dropbox.com.
[2] Amazon.com Inc., Amazon Cloud Drive, https://www.amazon.com/clouddrive/learnmore.
[3] Apple Corp., iCloud, http://www.apple.com/icloud.
[4] Canonical Inc. Ubuntu One, https://one.ubuntu.com.
[5] Apple Corp., Time Capsule, http://www.apple.com/timecapsule.
[6] Broadband Forum, http://www.broadband-forum.org.
[7] Broadband Forum,"TR-069 - CPE WAN Management Protocol specification v1.2, Amendment 3", November 2010.
[8] Microsoft Corp., "Microsoft SMB Protocol and CIFS Protocol Overview", March 2011.
[9] Postel, J. et al, File Transfer Protocol, IETF RFC 959, October 1985
[10] R. Fielding et al., "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC 2616, June 1999.
[11] Varia, J. "Amazon Web Services - Architecting for the cloud: best practices", White paper, January 2010.
[12] Broadband Forum, "Data Model Template for TR-069 Enabled Device, TR-106 Amendment 4", February 2010.
[13] Broadband Forum," Component Objects for CWMP, TR-157 Amendment 3", November 2010.
[14] Broadband Forum, "TR-069 Data Model for Storage Service Enabled Devices, TR-140, Issue 1.1", December 2007.
[15] IEEE, OUI Registration Authority, http://standards.ieee.org/develop/regauth/oui/public.html.
[16] Apache ActiveMQ project, http://activemq.apache.org.
[17] Git version control system, http://git-scm.com.
[18] Ylonen, T. et al, "The Secure Shell (SSH) Protocol Architecture", IETF RFC 4251, January 2006.
[19] Linux Kernel Documentation, "Why Not dnotify and Why inotify" http://www.kernel.org/pub/linux/kernel/people/rml/inotify/README.
[20] The thttpd project, http://acme.com/software/thttpd/
[21] The vsftpd project, https://security.appspot.com/vsftpd.html
[22] The Samba project, http://www.samba.org
[23] T. Cruz et al., "CWMP Extensions for Enhanced Management of Domestic Network Services", Proc. of LCN'2010 (The 35th IEEE Conf. on Local Computer Networks), Denver, USA, September 2010.
[24] M. Carbone, L. Rizzo, "Dummynet revisited", SIGCOMM CCR, Vol. 40, No. 2, November 2009.
[25] CACE Technologies, Wireshark Network Protocol Analyzer, http://www.wireshark.org.
[26] VMWare Inc., "Migrating to VMWare ESXi", White paper, 2011
[27] VMWare Inc., VMWare Virtual Networking Concepts Information Guide, revision 20070718.
[28] VMware Inc., VMWare VSphere 4.1 Networking Performance, Performance Study, April 2011.
[29] T. Cruz et al., "Integration of PXE-based Desktop Solutions into Broadband Access Networks", Proc. of CNSM'2010 (The 6th IEEE/IFIP Conf. on Network and Services Management), Niagara Falls, Canada, October 2010.
[30] Lorem ipsum generator, http://code.google.com/p/lorem-ipsum-generator.
[31] J. Kistler, M. Satyanarayanan, "Disconnected operation in the Coda file system", ACM Transactions on Computer Systems 10, February 1992.
[32] J. Howard et al, "Scale and performance in a distributed file system." ACM Transactions on Computer Systems 6, February 1988, pp. 51-81.
[33] P. Braam et al, "The InterMezzo File System", The Perl Conference 3, O'Reilly Open Source Convention, Monterrey, USA, August 1999.
[34] D. Terry et al, "The Case for Non-transparent Replication: Examples from Bayou", IEEE Data Engineering, December 1998, pp. 12-20.
[35] D. Peek, J. Flinn,"EnsemBlue: Integrating Distributed Storage and Consumer Electronics", 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06), Seattle, USA, November 2006.
[36] M. O'Keefe, P. Kennedy, "Enterprise data sharing with Red Hat Global File System", Red Hat Magazine issue 9, July 2005.
[37] Gluster Inc., "An Introduction to Scale-Out NAS for Cloud and Virtual Environments", Technical Whitepaper, March 2011.
[38] W. Yu et al, "Benefits of High Speed Interconnects to Cluster File Systems: A Case Study with Lustre", IEEE Parallel and Distributed Processing Symposium (IPDPS 2006), Rhodes Island, Greece, 2006.
[39] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, USA, December, 2004.
[40] S. Ghemawat, H. Gobioff, S. Leung, "The Google File System", The 19th ACM Symposium on Operating Systems Principles, Lake George, USA, USA October, 2003.
[41] Apache Hadoop Distributed File System, http://hadoop.apache.org/hdfs/
[42] B. Pierce, J. Vouillon,"What's in Unison? A formal specification and reference implementation of a file synchronizer", Tech. Rep. Technical Report MS-CIS-03-36, University of Pennsylvania, 2004.
[43] A. Tridgell, P. Mackerras, "The rsync algorithm.",Tech. Rep. TR-CS-96-05, Department of Computer Science, The Australian National University, Canberra, Australia, 1996.
[44] J. Li et al, "Router-supported Data Regeneration in Distributed Storage Systems," Poster at 8th USENIX Conference on File and Storage Technologies (FAST'10), San Jose, CA, February 2010.
[45] A. Karypidis, S. Lalis, "Omnistore: A system for ubiquitous personal storage management", In Proceedings of the 4th IEEE International Conference on Pervasive Computing And Communications, Pisa, Italy, March 2006, pp. 136–147.
[46] LaCie S.A., LaCie CloudBox, http://www.lacie.com/us/products/product.htm?id=10563.
[47] QNAP Systems, "Continuous Data Protection on QNAP NAS with ElephantDrive", Application note, http://www.qnap.com/index.php?lang=en&sn=2719