Cirrus: A Digitally Responsible Global Filesystem

William M. Pitts
Cirrus Project (DBA)
billpitts@stanfordalumni.edu

William J. Yeager
Retired, Stanford Knowledge Systems Lab
byeager@fastmail.fm

Abstract

Cirrus is a distributed filesystem, overlay network that extends the service domain of file servers to global scale without diminishing the quality of service. Cirrus, developed over many years, is operational today and is ready for testing and benchmarking. Cirrus' distributed shared memory implementation provides a fast and secure method of transporting all network traffic within the overlay network.

1. Introduction

Since the introduction of the World Wide Web in the early 1990's the Internet has evolved in a rather haphazard manner with too many organizations and individuals contributing extensions and new network protocols with "Informational Status" that have not been vetted with a rigorous IETF review [1].

The Internet's "lower half", developed from the mid 1970's to the late 1980's, delivered the foundations of DNS, IP, UDP, TCP and the supporting protocols that made it all work. Each foundational protocol is the result of a thorough, multi-company, IETF design and review process followed by an implementation by the principal designers.

When Tim Berners-Lee opened the door to WWW development, many new protocols were hastily developed as companies raced to stake their claims in the new frontier. This process produced an Internet "upper half" built on a proliferation on non-vetted protocols yielding a weak as well as vulnerable layer.

Today's Internet delivers benefits that enrich and streamline our lives, enabling us to do more with little expended effort. Expanding our horizons to global scale, we are able to easily shop in the markets of foreign countries, to maintain contact with friends as they travel, and to collaborate with distant colleagues. The benefits are substantial, as evidenced by the fact that most of us opt for phones that double as personal Internet portals (that we wouldn't leave home without!). However, today's Internet also exhibits many deficiencies and flaws. The most obvious include an abundance of security holes, weak consistency and poor performance.

1.1 Shortcomings and flaws

Network congestion, often trumping satisfaction with frustration, is most apparent when a streaming video or audio program comes to a screeching halt. In these cases, the user, monitoring a real-time data stream, becomes instantly aware of a network shortcoming. However, in many instances network congestion and extended response time latencies remain hidden from the user. But these network delays are readily apparent to distributed applications and often substantially limit the scalability of such applications. Consider that the NFS and CIFS distributed filesystems can only support clients within a campus sized geographical area. Although geographically remote clients may access an NFS or CIFS file server, it is rarely done because the performance is so poor.

Another shortcoming ISPs and system administrators wrestle with is the excessive complexity of managing an infrastructure composed of major software components from many vendors. First, there are multiple operating systems and their built-in protocol stacks running on various hardware platforms. Then, application software, layered above the operating systems, provides services such as NFS, CIFS, AFP, DNS, LDAP, NIS, FTP, NTP, RDP, RIP, RTMP, RTMFP, SIP, SLP, SMTP, IMAP, SSH, etcetera. The misconfiguration of any one of these can create a security hole. This security vulnerability is further compounded by the fact that all of the aforementioned software is in a constant state of flux because the various vendors are continually updating their products.

Stepping up from *shortcomings* to *flaws*: security holes abound within the Internet. Reports of data breaches within corporations and government agencies occur on an all-to-often basis. Identity theft and ransom ware attacks are growing, putting all of us at risk. We are aware of the problems. Yet, while countermeasures are continuously deployed, these attacks are still on the increase [2]. So, we keep our fingers crossed and hope Internet evolution plugs the holes before we get stung.

1.2 A Bit of history

These Internet deficiencies trace back to the earliest days of the development of the World Wide Web (early 1990s). At that time the capabilities of both the NFS and CIFS distributed filesystems limited operations to clients geographically close to the file server (on the same campus) [3][4].

When Tim Berners-Lee created the first truly global file sharing system, he could not call upon an operating system's filesystem to deliver file data to very remote clients. So, he did what all good hackers do, he did it himself! And, he did it in user space.

The CERN httpd web server executed as a user space daemon. It responded to http browser requests by calling upon the host's filesystem to retrieve the appropriate file data and then sent that data back over the Internet to the browser. This client-server program launched the Web.

Tim had developed a distributed hypertext system that made published material very accessible. I suspect this was his goal. Perhaps he envisioned far more. Perhaps he envisioned the Internet as the global communications and commerce infrastructure it has become. However, today Tim is concerned about the Internet's current state: 30 years on, what's next #ForTheWeb? [a]

From the perspective of viewing the Internet as it exists today, a major architectural flaw was introduced at the point where the filesystem passes file data back to a user space web server, and at that point the web server assumes responsibility for the global distribution of the file data. Filesystems are very good at providing secure file access only to authenticated users. But, when a filesystem delivers file data to a web server, it relinquishes all control over that data.

Regardless, Tim's effort started the ball rolling. The Web quickly became a powerful advertising medium, which led to sales, which led to secure transactions, and on and on to where the Internet is today.

The foundations of the Internet standards (primarily IP, UDP and TCP) enable any developer to create new distributed application communication protocols with relative ease. So, as the Internet quickly evolved to provide an expanding array of services, the thousands of developers working at hundreds of companies were not hindered from creating additional network protocols to support new Internet applications.

This wide-open development process quickly delivered a global Internet that delivers substantial benefits to all. Unfortunately, it has the shortcomings and flaws mentioned above. And these are difficult to fix. In particular, it is not possible to secure an extremely complex Internet when so many of its components continue to produce zero-day security vulnerabilities. Furthermore, the integrity and allegiance of maintenance programmers and system administrators distributed about the globe cannot be ascertained or guaranteed. There are just too many doors and too many locksmiths!

In summary, the Internet's major faults are:

- Too many doors and too many locksmiths.
- Filesystems prematurely relinquishing control over content to web servers.

1.3 However, there is a solution

A distributed filesystem overlay network provides an ideal framework for operating in parallel with the vulnerable layer of user level applications. The overlay network rests upon a minimal set of pillars anchored in the bedrock of TCP, IP, DNS and LDAP, that is to say, what's required to address, send and receive messages.

The framework of a distributed filesystem provides the essential ingredients required to construct a fast, secure, consistent and highly available global network.

Note the transition from *distributed filesystem* to *global network* in the preceding paragraph. When the distributed filesystem's consistency mechanism is capable of delivering *sequential consistency* [b] very quickly, then distributed shared memory (DSM) becomes a very effective method of communicating across a network [5][6][7][8][9].

When the global network employs DSM for all network communications, the "too many doors" issue is addressed. There is only one door: *the filesystem API*. And this door is fitted with a *virtualized locking mechanism* that can be configured on demand. Every object (and every directory leading to the object) is allowed to specify the door lock used to secure content. Locks can be developed in-house, purchased from a third party, or selected from an assortment of standard built-in locks.

There might still be "many locksmiths" (especially if a sizeable third-party market develops for locks), but not "too many" because the content owner or system administrator is able to select locks that are well vetted and trusted. And, for the paranoid, the door can be configured with multiple locks.

However, fixing the Internet's second major fault is a more daunting task. More than thirty years after the introduction of the first version of NFS, both NFS and CIFS file servers are still "stuck on campus". Stretching the geographic domain of a distributed filesystem to operate at a global scale will require:

- A more efficient network protocol for projecting consistent file data into geographically remote sites.
- Hierarchies of intermediate file caching sites that provide global access to file data at "local" speeds.
- A hierarchical distributed consistency mechanism that ensures the consistency of file images throughout the file caching hierarchy.
- High availability capabilities based on the hierarchical distributed consistency mechanism and on redundant network links to provide continuous file access as long as any operational path exists between a client node and an origin server.

A global distributed filesystem with these features and capabilities, projecting consistent file images into edge nodes, services file access requests at LAN speeds. It is a superhighway for the Internet.

The filesystem API provides portals to the superhighway for application programs, and is the sole interface for both requesting file access and for sourcing file data. The superhighway is middleware: the same API that enables an application to request file data is employed at the other end of the superhighway to request the file data from an origin file server.

Existing programs (MS Word, for example) will not be able to distinguish this global filesystem from their "traditional" local filesystems. So, these programs can now directly locate and open any document, anywhere, without the assistance of any helper program. API extensions will be required to exploit new filesystem capabilities, but these extensions will not disrupt the operation of unmodified application programs.

The superhighway is an overlay network. It will not prevent any parts of the "old" Internet from operating as before. Think of the "old" Internet as the road network that existed in America in the 1940s, and the superhighway as the Interstate Highway System developed in the 1950s and later. The old roads are still available, but it makes a lot more sense to use an interstate highway for a long trip.

So, as the superhighway is deployed in stages and grows in performance and capabilities, application developers and maintainers are free to choose when and where to merge onto the superhighway. They will merge because, at some point, the reasons become compelling.

The superhighway has not been built to date because it requires an enormous amount of distributed state information be kept synchronized, and this is an extremely daunting task (see *Consistency and Availability on a Global Scale* [c]). Furthermore, the companies developing operating systems, Internet software and application programs all have a very parochial view; only considering efforts in their marketplace can that increase their revenues in the short term.

The field is wide open for a dedicated, very talented, multi-discipline team to create the superhighway that will transform the Internet. And, to aid in that effort, there is a running head start: the Cirrus Global Network.

2. Cirrus Global Network

Cirrus, an enormous geographically distributed filesystem, envelopes the shared content of all file servers worldwide into a common namespace and provides highly available, consistent file access services to all content. Wherever you travel in the world, your data (and all other data) is *here*.

Cirrus is both a geographically distributed virtual file server and a very fast data communications overlay network. It is a superhighway system for the Internet. Its freeways are fast, and enormous fully automated warehouses may be placed at every intersection and off-ramp.

A Cirrus node stores an image of every file that transits the site in its warehouse. Later, when servicing a file access request, the Cirrus node first checks its warehouse to determine if a current image of the target file is present and, if so, uses that file image to respond to the request. As a consequence, file data retrieval is deceptively fast because fetching data from warehouse DRAM or flash memory (or even disk) is much faster than retrieving the same data across a high latency WAN link [10][11][12][13].

A distributed consistency mechanism interconnects all warehouses, enabling each warehouse to ensure it never references a file image not consistent with the most recent version of the file. Furthermore, the Cirrus distributed consistency mechanism transparently overcomes network node and link failures as required to deliver absolute file consistency. Neither the client nor the server need be made aware of the failure. They are informed only when Cirrus cannot re-route around the failed component(s). And this only happens when the network has not been configured with sufficient redundancy.

Cirrus is an overlay network, meaning it is layered on top of the existing network infrastructure. The underlying Internet layers are still there and can be used as before. And the Cirrus overlay network is not very crowded because all repeat traffic has been removed.

A Cirrus node is a networked computing device configured with the Cirrus code module. The device may be a phone, laptop, computer, file server, or switch. The Cirrus code is essentially the same for all devices. When a Cirrus node initializes, Cirrus configures itself appropriately for the host device. A Cirrus phone might initialize with a 512 Gb warehouse, whereas a backbone node could configure itself with a two-stage warehouse of 4 Tb DRAM and 64 Tb of flash/disk. Every file that has recently passed through the node would be immediately accessible in DRAM, and files referenced in the last three months may still be found in the flash or disk portions of the warehouse.

The warehouse within a Cirrus phone operates in the same manner. But, since the phone's owner has requested all file data passing through this edge device, this warehouse ends up containing the 512 gigabytes of data most recently referenced through this phone. Note that the user's home folder is origined on some big Cirrus file server somewhere out there. This is the only location where the user's data resides. It may be of any size. It will always be accessible, and it will never be lost. The 512 gigabytes in the phone's warehouse is the most recently referenced file data, regardless of source: web surfing or home folder.

When the phone is misplaced, its warehouse may be instructed to freeze or to delete all content. When the user finds the phone or initializes a new one, the warehouse simply refills itself by operating as normal. All of the user's devices "see" the same home folder and can access any part of it. All user devices remain synchronized at all times. That's just the way it works.

Cirrus nodes communicate as peers. Every node is an access portal and every node may be an origin server, "owning" and managing the content it provides to the Cirrus network. However, the role of origin server is usually left to big file servers deep within the Cirrus network. These systems are configured with substantial redundancy and are managed to ensure their files are never lost and are always accessible.

The phone's main memory (DRAM) allocated to the warehouse is Cirrus Global Network memory. Its contents may be quickly accessed by the phone's user or applications, but that memory is owned and controlled by Cirrus. The warehouse is the phone's exclusive high-performance portal into the Cirrus network. In fact, the phone is part of the network. After proper authentication, the phone's warehouse memory may be directly accessed via other Cirrus nodes using the full set of filesystem API routines, including memory mapped i/o. Multiple devices scattered across the globe directly reading and writing the same memory. *This* is the Internet of Things!

2.1 Cirrus structure

Cirrus is comprised of millions of *crystals* (file service proxy cache nodes). A crystal is a network node with compute and storage resources configured with the *Crystal Module*. Servers, gateways, computers, laptops and phones may be crystal configured, thereby weaving them into the Cirrus network.

Every crystal contains a massive warehouse filled with the content that has most recently passed through the site. Cirrus uses this content to respond quickly to network requests, sidestepping the need to request data from origin servers. Warehouse content inconsistent with its source is detected and discarded before its use. Today's enormous DRAM and disk capacities allow content to "age" for weeks or even months before it must be discarded.

Nearly identical Crystal Modules run efficiently across the spectrum of hardware platforms. Picture the Cirrus distribution system as a network of huge pipes interconnecting servers within a rack, big pipes connecting gateways and other servers in the room, medium pipes (WAN links) connecting remote intermediate crystals, and finally medium or small pipes to edge devices.

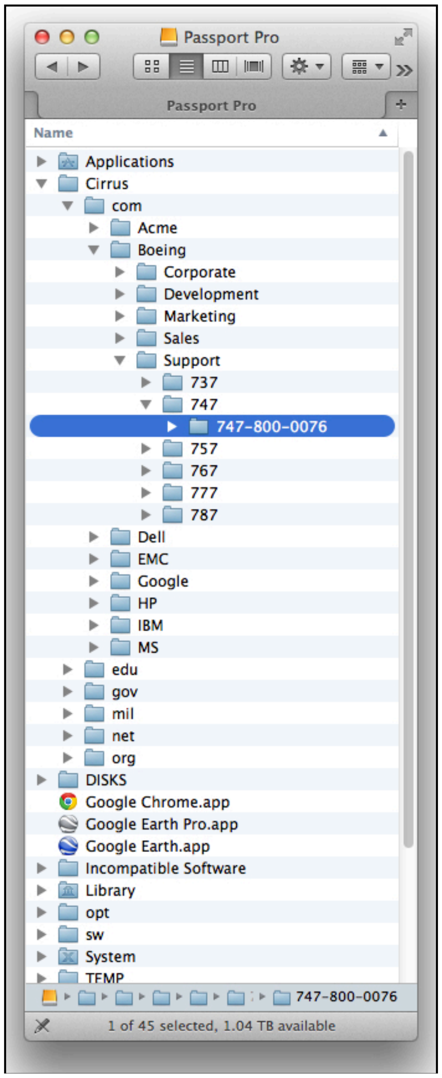
Pipe sizes are dynamically scaled throughout the distribution network based on the capabilities of the network link, the size and type of file and the current loading of the server-side endpoint. Endpoint crystals negotiate pipe diameters each time a file connection is established, much like Ethernet endpoints may negotiate the MTU that is used between them. Then a uniform distributed control mechanism continuously monitors and directs traffic throughout the distribution fabric.

All crystals incorporate both server and portal capabilities. However, edge and intermediate crystals may be restricted to providing file access services (the portal role). A server crystal is an origin server for at least some of the content it provides.

Crystals bind together in a recursive manner to create virtual file servers that are then incorporated into successively larger virtual file servers. At the top there is but one server, the Cirrus Global Filesystem.

NFS and CIFS file servers can also be Cirrus edge devices, using NFS or CIFS protocols to project Cirrus content over the "last mile" to unmodified NFS and CIFS clients.

The filesystem depicted below shows the top-level directories of the root disk of a Mac desktop computer.


```
Passport Pro
Passport Pro
Name
▶ Applications
▼ Cirrus
    ▼ com
        ▶ Acme
        ▼ Boeing
            ▶ Corporate
            ▶ Development
            ▶ Marketing
            ▶ Sales
            ▼ Support
                ▶ 737
                ▼ 747
                    ▶ 747-800-0076
                ▶ 757
                ▶ 767
                ▶ 777
                ▶ 787
        ▶ Dell
        ▶ EMC
        ▶ Google
        ▶ HP
        ▶ IBM
        ▶ MS
    ▶ edu
    ▶ gov
    ▶ mil
    ▶ net
    ▶ org
▶ DISKS
  Google Chrome.app
  Google Earth Pro.app
  Google Earth.app
▶ Incompatible Software
▶ Library
▶ opt
▶ sw
▶ System
▶ TEMP
747-800-0076
1 of 45 selected, 1.04 TB available
```

/Cirrus/com/Boeing is the path to the Boeing server. This server, containing all of Boeing's online documents (public and private), is a geographically distributed multi-homed virtual file server. It may have thousands of access portals around the world, relying on GeoDNS to direct access traffic to portals close to users. All portals provide fast, consistent read/write access to any document. Of course, a thoroughly tested authentication mechanism stringently controls what is accessible, or even visible, to users.

/Cirrus/com/Boeing/Support/747/747-800-0076 is the path to the folder containing all engineering and maintenance documents related to a 747-800 aircraft with serial number *0076*. No matter where this aircraft travels, its complete documentation set will be available for reference and updating.

Higher-level Cirrus directories are forgetful by design. When the */Cirrus/com* directory is entered, only the thirty most recently referenced sub-directories are presented.

If a computer with the directory view depicted were resolving the path */Cirrus/com/Apple/Support/iPod.doc*, it would fail to find *Apple* in the *com* directory. But, since *com* is forgetful, there is a means to remember.

com/Apple is parsed and *Apple* is resolved to an IP address (DNS or LDAP) and a request is sent to the virtual file server at that address to connect to the root of its export tree. Finally, "*Apple*" replaces the least recently used directory in the *com* folder and its contents may be discovered and accessed.

Within Cirrus, folders scale from directories with a few files to huge domains containing content from around the globe. The higher-level folders are actually domains and may be configured to perform domain manager type functions. The terms "folder" and "domain" may be used interchangeably, but "domain" generally implies some administrative control is being applied to inbound and/or outbound network traffic.

2.2 Centralized control over distributed data

Two inviolate Cirrus principles are:

- *Cirrus domain managers may exercise control over all content within their domain throughout the Cirrus global distribution network.*
- *Cirrus nodes are trustworthy.*

These two principles provide the foundation for a global filesystem that promiscuously caches file data all over the world while ensuring content owners retain complete control over their content.

Higher-level domains cannot remove restrictions or controls placed on content by lower-level domains; only additional restrictions or controls can be placed on content. So, file access requests may be required to run a gauntlet of challenges before getting to sample any file data.

Owners of higher-level domains do not necessarily own content within their sub-domains. And yet, they may still exercise control over any content within their domain.

Control over content is exercised by attaching *policy attributes* to content in the form of extended attributes. The consistency mechanism ensures all sites know the policies, and all sites are faithful and trustworthy.

Such an arrangement can be used to guarantee the adherence to contractual, as well as complex DRM models.

2.3 Extensibility

Cirrus is extremely extensible. For example, when a domain manager specifies a third party developed authentication module is to be used for authentication, Cirrus will automatically load that module at remote sites whenever necessary.

A major design and implementation goal is for Cirrus to be as extensible as possible. Only the most minimal framework and the distributed consistency mechanism should be static. Replaceable modules should eventually encompass network transfer protocols, encryption/decryption modules, authentication modules and presentation modules.

2.4 Security

Cirrus addresses network security in a manner designed to satisfy the concerns of individuals, corporations and governments. Recognizing that back door fears will always be present with any Cirrus controlled solution, Cirrus' extensibility features provide organizations with methods to secure their data from end-to-end. Corporations, for example, may use encrypted filesystems (in-house developed or purchased from third party) and instruct remote sites to use the corresponding authentication and decryption modules. This approach removes any possibility of back doors into unencrypted file data. Only encrypted file data flows beyond the origin filesystem. Data security is owned by the corporation's IT department, as it should be.

Third party developed software will quickly make this level of security available to individuals. And, at that point, governments around the world may decide that encrypted communications over the Internet are only permitted between authenticated parties whose identities are verified, for example, by iris scan, voice print, facial recognition, thumbprints or suitable combinations of these. Thus, unauthenticated encrypted traffic travelling on the Internet could be detected and blocked. This will make it far more difficult for those with nefarious intentions to communicate.

Top-level domains such as */Cirrus/com* are public. Public domain portals do not usually exercise much control over inbound/outbound traffic, but they may if a need to do so is identified (such as blocking encrypted network traffic). However, */Cirrus/com* sub-domains are the virtual file servers of ISPs, organizations, corporations and governments. The domain managers of these domains enforce a large set of complex policies ensuring "outsiders" only access content meant for public distribution while allowing "insiders" to roam deeper into the domain's filesystem and to access and modify content as permitted by their credentials.

Furthermore, the portals of these domains are transition points from "inside" to "outside". So, this is often the boundary where file data is encrypted and policy attributes are attached to the file.

2.5 Simplified administration

GUI based file managers render the configuration and management of domains and domain managers straightforward and intuitive. Right clicking on any folder presents a list of the "doable" (in black) interspersed with the "not doable" (in gray). Selecting a complex "doable" feature may present an appropriate form for the administrator to complete. Placing the cursor over any of the "not doable" will pop-up an explanation of why that feature is not available or currently not possible.

This procedure is used throughout the domain/folder hierarchy and all features and capabilities are available at every level unless enabling a particular feature just does not make sense. This implies every folder is a domain and, as such, it may exercise control over its content. So, if desired, a folder near the bottom of a filesystem hierarchy may be configured to demand extra security measures be applied throughout the distribution network when transporting, caching or accessing its content.

Cirrus simplifies administration and management tasks by removing as many distinctions as possible between the levels of a global filesystem hierarchy. It is easier to comprehend one container type than three, four or five container types. If this is not readily apparent, then please go online and check out Microsoft's Active Directory documentation.

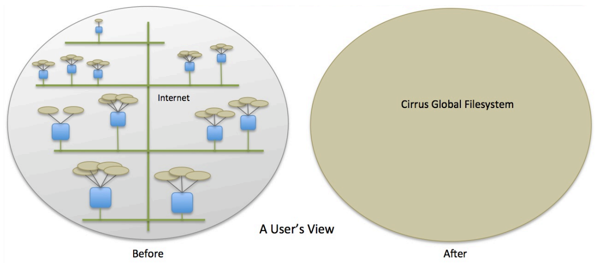
Moreover, the GUI listing of black and gray features (with pop-up explanations) assists users and administrators in understanding the capabilities of a domain/folder. While experienced system administrators will be managing higher-level domains, individuals with little or no technical training will be administering their small part of the Cirrus Global Filesystem. Therefore, the simplicity of this user interface is quite important.

3. New perspectives on a global network

Cirrus transforms the Internet's exterior, fundamentally transforming the Internet's appearance and the methods by which distributed applications communicate.

3.1 The user's view

Cirrus presents both users and developers with a new view. Starting with the user's view, the figures below depict the Internet *before* and *after* the full deployment of the Cirrus Global Filesystem.



The figure on the left presents a view of the Internet as containing a number of disjoint filesystems (tan) attached to computers (blue) sharing a common network.

The figure on the right shows a single filesystem. The disjoint filesystems of *before* have been stitched together into a single filesystem with a global namespace.

Web browsers are designed to mask the complexity of leaping across the Internet from reference to reference. Consequently, for Web surfers the new Internet looks a lot like the old. The only visible difference is the browser's URL bar displays URLs starting with "file://" instead of "http://". (Browsers require no modifications; they already operate in this manner.)

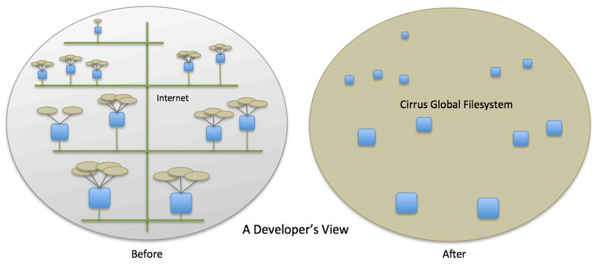
A more important difference, not immediately visible, is all referenced content is completely current. Content modifications performed anywhere prior to a reference are included in the pages displayed for that reference. For companies engaged in Internet commerce, consistency of content creates many opportunities to improve their online business practices. Furthermore, the new Internet offers a more secure and a more scalable content distribution network.

For computer users, the "reach" of application programs originally developed to execute on local filesystems (Word, Acrobat, Photoshop, …) is now global scale. MS Word, for example, can directly open and edit any document anywhere (permissions permitting, of course). The world's complete filesystem, contained within the Cirrus folder, is now within the reach of applications developed before there was an Internet.

In summary, the user's view does not change. It just gets better and far more expansive.

3.2 The developer's view

The developer's view of the Internet also gets better, but more dramatically. The figures below graphically show the fundamental nature of the transformation wrought by Cirrus.



The *after* view shows a single filesystem, the same set of computers and no network! Instead, the computers are encompassed by a global filesystem. According to this view, if the computers are to communicate it must be through the filesystem.

Before the time of networked computing (before Ethernet), timesharing systems were the "network". The larger timesharing systems could support hundreds of simultaneous users. When users collaborated while online, the timesharing system was their network.

Applications and processes executing on the same system communicated using inter-process communication (IPC) mechanisms such as signals, pipes, semaphores, sockets and shared memory. All of these IPC mechanisms are in common use today.

With shared memory, two or more processes map the same memory into their address space. Each process executes a single system call instructing the operating system to perform the mapping. Afterwards, whenever any process modifies the shared memory, that modification is immediately visible to all other processes (if they bother to look).

No interaction with the host operating system is required after the first system call.

However, when a process modifies shared memory a signal is often used to quickly alert the other processes. Otherwise, it might be a while before the other processes notice the change. Sending the signal does require a system call.

Cirrus builds upon the POSIX version of shared memory, where the shared memory is a file residing within a filesystem. And Cirrus extends the filesystem interface, enabling the fast delivery of notification messages to distributed processes whenever their shared memory is modified (US Patent 8,504,597, others pending). With Cirrus, writing to shared memory incorporates the ability to send a signal to other processes.

Distributed applications may use Cirrus' extended version of POSIX shared memory to communicate quickly and efficiently. Distributed applications open the same file for read and write access and then map the file as shared memory. Now the applications can exchange messages and data. The distributed application is up and running and a file name was the only link required to bind the applications together.

The distributed applications running on the Internet today are excessively complicated because when an application "gets distributed", at least some if not all of the networking fundamentals mentioned a few paragraphs earlier come into play. That is a lot of complexity. With Cirrus, all of that complexity is still there. But Cirrus layers on a view and an interface that hides the complexity not directly related to the distributed application.

3.2 Beyond views

The streamlined views Cirrus presents to users and developers is a major improvement over the current Internet. But this improvement is not substantial enough to warrant the use of the Cirrus architecture. That justification rests on major improvements in performance, scalability, availability and security.

The Cirrus overlay network of millions of crystals (huge DRAM/disk caches with strong consistency) securely and intelligently transport and cache file data on a demand basis. The same content never traverses a network link twice (*never* being defined as "at least a week, but possibly forever"). With repeat traffic removed from the network and content cached close to access points, the existing infrastructure will deliver far better performance to substantially more users.

Cirrus is a superhighway overlay for the Internet. Every off-ramp may contain a huge, fully automated warehouse containing the content that has most recently passed through that site. And warehouses never serve up stale content. This is the *fully automated* part.

DRAM, flash and disk capacities continue to increase while their prices decline, and non-volatile main memories are on the horizon. Meanwhile the cost of WAN links, in terms of both price and increased response time latencies, remains high. So, it may often be very cost effective to place warehouses configured with a terabyte of DRAM and twenty terabytes of disk at WAN off-ramps.

It is also very effective to locate warehouses in user devices such as phones, pads and laptops. A 512 GB phone, for example, might allocate 256 gigabytes for its internal warehouse. This might seem to be a rather excessive allocation, but the warehouse replaces the phone's internal filesystem. The user's content resides in the Cirrus cloud and is accessible and shared by all of the user's devices.

Warehouses may also be preferentially located to minimize Internet backbone traffic when retrieving data to unburden border routers and gateways. In the mature stages of deployment, the Cirrus network will contain millions of warehouses. Content will be stored all along the pathways through which it has been previously accessed. Origin server content is projected throughout the cloud and to its edges, close to where the content is being accessed. Often, the final warehouse is in a user's device.

4. Digital rights and responsibilities

Cirrus' security capabilities (Section 2.4) and its ability to exercise centralized control over distributed data (Section 2.2) endow Cirrus with facilities for protecting individuals and safeguarding personal data.

The Cirrus Global Network can be configured to:

1. Reflect that computer technology is a means of amplifying an individual's mind. Encrypted personal data stored online may be considered an extension of an individual's personal thoughts and be subject to Fourth Amendment protections.
2. Store browsing history, search history and other online activity histories in folders that belong to, and are controlled by, the individual. The individual may choose what is disclosed, when it is disclosed and whether any payment is required.
3. Not allow anonymous users to post articles or send email.
4. Proactively detect and prevent many types of malicious activity.

5. Cirrus today

The current Cirrus implementation is a Linux user space application written in C. The core functionality required to benchmark and verify global scale fast access to consistent file data is operational. The high availability capability (failover) is partially implemented, but the design is complete. Eight US patents have been granted and nine US patent applications are being prosecuted.

Cirrus configured Amazon machine images (AMIs) are available in the Asia Pacific and US West regions of the AWS cloud. Preliminary benchmarking shows a very substantial performance advantage of Cirrus over NFSv4 when the "ping latency" between nodes averages 125 milliseconds.

Three zip files are unzipped and directories, sub-directories and files are created in exported filesystems resident on a server in Tokyo. Then the *configure* script (one of the files within the zip file) is executed and it generates substantial filesystem traffic as it generates a configuration file that controls the "make" process. This "make" process compiles source code files and creates object code files, and then it links the object code files and creates an executable program.

	Cirrus	NFSv4
gz124src.zip 270kb		
unzip	6.07 secs	83.56 secs
configure	0.60 secs	55.62 secs
make	1.07 secs	37.05 secs
dds.tar.gz 512kb		
unzip ("tar xvf")	6.87 secs	50.28 secs
configure	8.07 secs	633.73 secs
make	20.46 secs	247.65 secs
tar-1.28.tar.gz 4mb		
unzip ("tar xvf")	32.69 secs	702.45 secs
configure	42.08 secs	<-broken pipe to tk
make	24.05 secs	after ~ 2500 secs

Performance Comparison: Cirrus vs NFSv4
Tokyo Server and N. California Client

The important numbers to note are the extremely fast Cirrus execution times in each phase of building an executable program from a zip file residing on a very remote file server. Cirrus achieves this performance level by leveraging its distributed consistency mechanism to drastically eliminate network traffic while still maintaining POSIX filesystem semantics. The Cirrus and NFSv4 servers both executed on the same AWS instance at the same time. All other aspects of the comparison runs were fair and equal. Detailed information on these comparison runs is presented at http://www.billpitts.org/Cirrus_vs_NFSv4 [d].

Cirrus' incredible performance stems from its distributed consistency mechanism. After unzip (or tar) creates the root directory for the files and directories being unpacked, no other communication with the server is required by the client system. However, the root directory and its contents are contained within a filesystem exported by the server and may be recalled by the server at any time.

The time has arrived for Cirrus' transition from the development lab to the marketplace. A substantial effort will be required to field a first Cirrus product, which has not been determined at this point. However, here are several potential candidates:

- Deploy Cirrus file servers in Amazon, Google or Azure clouds to invert the online backup business. Your data does not reside on a computer at home and need to be secured by Carbonite. It resides securely in the cloud and is projected into all of your devices. You can modify content on any device, and that content instantly changes on all devices. You can lose any device and not lose any of your data. This is a pure software play.
- Launch a new file server company to compete against NetApp, EMC, Dell and others. Very late to this game, but geographically distributed file servers binding together to create a single global, highly available consistent filesystem is a compelling feature. This could be a mostly software play with Cirrus servers running in cloud and Cirrus access portals running on real hardware on clients' premises.
- Start a company that develops and licenses global filesystem software, Veritas style.

There are many promising avenues to consider. Cirrus could become an opensource effort or it could develop along a proprietary path. No decision has been made at this point because, in all honesty, there is no plan. A next step is to assemble the best possible management team. Their first task will be to identify the first Cirrus product and develop a plan for developing and marketing it.

Meanwhile, Cirrus development activities will continue. If you would like more information on Cirrus and its current state of development see: http://www.billpitts.org/Cirrus.[e](Password: *Cirrus*)

6. References

[1] Bradner, S., RFC2026, IETF, October, 1996.
[2] Information Week, Dark Reading (https://www.darkreading.com/) [f]
[3] Haynes, E., RFC7530, IETF, March, 2015.
[4] Neville-Neil, G., "Bound by the Speed of Light", ACM Queue, Vol 8, Number 12.
[5] Khakidi Y., and M. Nelson, "The Spring Virtual Memory System", Sun Microsystems Laboratories, SMLI-TR-93-09.
[6] Cheriton, D., "The V Distributed System", Communications of the ACM, Vol 31 Number 3 March 1988.
[7] Young, M., "Exporting a User Interface to Memory Management from a Communication-Oriented Operating System", Technical Report, CMU-CS-89-202, Carnegie Mellon University, November 1989.
[8] Rashid, R., A.Tevanian, M. Young, D. Golub, R. Baron, D. Black, W. Bolosky, and J. Chew, "Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures", *IEEE Transactions on Computers*, Vol 37 August 1988.
[9] Abrosimov, V., F. Armand, and M. Ines Ortega, "A Distributed Consistency Server for the CHORUS System", *3rd Symposium on Experiences with Distributed Multiprocessor Systems*, SEDMS III pp. 129-150, March 1992.
[10] Pitts, W., "System for accessing distributed data cache channel at each network node to pass requests and data", US Patent 5,611,049.
[11] Alonso, R. and M. Blaze, "Long-term Caching Strategies for Very Large Distributed File Systems", *Proceedings of the USENIX Summer Conference*, pages 3 – 16, June 1991.
[12] Alonso, R. and M. Blaze, "Dynamic Hierarchical Caching for Large-scale Distributed File Systems", *Proceedings of the Twelfth International Conference on Distributed Computing Systems*, June 1992.
[13] Chankhunthod, A., P. Danzig, "A hierarchical Internet Object Cache", University of Colorado, Boulder, *Computer Science Technical Reports,* CU-CS-766-95.

7. Hyperlinks

[a] https://webfoundation.org/2019/03/web-birthday-30/
[b] https://en.wikipedia.org/wiki/Sequential_consistency
[c] http://www.billpitts.org/Global_Consistency
[d] http://www.billpitts.org/Cirrus_vs_NFSv4
[e] http://www.billpitts.org/Cirrus
[f] https://www.darkreading.com/