Decentralizing the Internet

Adriana Flores, Clayton Shepard, Ellis Giles, Haihua Shen, Yanda Lu
Rice University

***Abstract*—We propose to build and investigate a novel decentralized internet architecture in which servers are replaced with point to point communication for personal communication and services. Previously, servers have been a necessary part of the Internet to act as intermediaries between people who didn't have pervasive connectivity. With the wide-spread adoption of smartphones and wireless data, this antiquated paradigm is becoming less and less necessary, since there is no technological reason that devices can't connect directly to one another. Indeed, there are many potential advantages to decentralizing the Internet, including security, privacy, resilience, cost, and power. However, this novel decentralized architecture also presents many challenges, such as redundancy, uptime, and failover. In this course project we will build a first generation decentralized Internet framework and address some of these challenges, as well as explore the unique opportunities, presented by this architecture.**

I. Introduction

In the early days of the Internet it was incredibly rare for normal users to have persistent, always-on connections. This limitation was addressed by servers, computers that were always on and always connected to the Internet, which served as an intermediary for communication between users. This server-client paradigm shaped the architecture of the Internet, and still dominates web communication today.

However, with the penetration of always-on broadband and data-enabled cell-phones, coupled with the emergence of IPv6, which includes mobile IP, there is no longer a technological limitation which requires servers. The computers in our pockets have over a 99% uptime and connectivity [1]! Indeed, centralized servers have many drawbacks, ranging from privacy, security, and fault-tolerance, to even energy-efficiency and control over data. An example centralized system is shown in Figure 1. By decentralizing the Internet we could make it much robust to failure, much more secure; furthermore we could do away with many energy-hungry server-farms, and reclaim control of personal data. An example decentralized system is shown in Figure 2.

While the idea of using our phones as personal servers is rather straightforward, it still raises many technical barriers. How do we ensure reachability? What happens if the cell-phone loses connectivity? What if it is lost/broken/stolen? We need to find novel methods to ensure redundancy for both uptime and data. To address these issues we propose building a small cluster consisting of your personal devices (home computer, laptop, even router) possibly combined with friends and family devices to guarantee reliability and connectivity.

In light of this, we propose to build a first generation decentralized architecture which supports typical personal communication and applications including:

- Email
- Web Pages
- Social Networking
- Chat

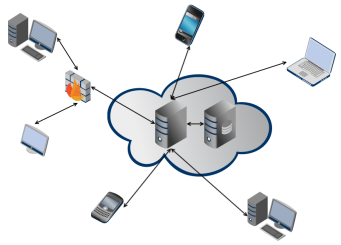

Fig. 1. Centralized internet architecture.

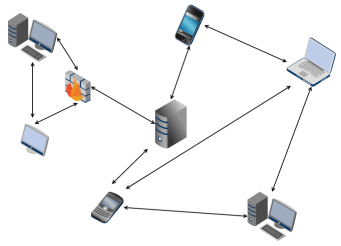

Fig. 2. Decentralized internet architecture.

II. Related Work

A. Decentralized Internet

Decentralized Internet is a hot topic for network researchers. From the application perspective, centralization and decentralization each has advantages and disadvantages [2]. Some of the more recent works have been done to provide alternative decentralized solutions for online social networks. For example, [3] proposed a decentralized version of Facebook to enhance privacy, and [4] suggested a P2P version of Twitter to improve scalability and reliability. In January 2014, a new project called Bitcloud was launched by the Bitcoin developers, seeking the possibility to decentralize the current Internet in order to improve privacy, security, and solve other current problem (like censorship) in today's centralized network. Our goal is to investigate the possibility to use our phone as decentralized storage to support applications in a decentralized network. To achieve our goal, we would need to look into a variety of issues, such as the trustworthiness of storage, the reliability of communication, and the ability to recover from single-device failures.

B. Data Replication

Data replication and consistency are important aspects of any system where the availability and reliability of systems services are important. Additional concerns such as scalability, performance, and fault tolerance play a role in deciding the best methods for data replication. Data may be replicated at either file or data block levels, each presenting a unique set of challenges. Considerations such as the network partition size, update frequency, replication granularity, number of users, security, reliability, and mobility all affect choices on the data replication policy.

Data replication in distributed systems, such as a Peer-To-Peer systems, where nodes are unreliable, additional considerations for the dynamic nature of nodes and data replication must be considered [5]. Plover [6] is a low-overhead file replication scheme for P2P networks that makes copies of files among physically close nodes based on capacities. Decentralized P2P file sharing programs such as Limewire, Bittorrent, Kazaa, etc. replicate file data on end nodes, which may become corrupt or suffer from loss of availability.

Replica files are copies of a master file in a file replication system. Files are replicated along data paths to reduce hot spots and improve efficiency. A file replication algorithm that achieves high query efficiency at a low query cost for decentralized file sharing systems has been developed [7]. Demand based file replication and consistency by means of a group of File Replication Servers (FRS) [8] has been explored which push data to other FRS when a requesting node requests a file. PAST [9] and CFS [10], a wide-area Cooperative File System, replicate files on nodes close to the owner of the file. While PAST stores whole files, CFS replicates data at the block level to distribute the load and storage space among servers in the network. CFS also uses Chord [11], a scalable P2P lookup protocol for Internet Applications, to maintain routing tables used to find blocks.

Web applications often benefit from data replication of application data by low latency access and reduced network traffic. Edge service delivery of content such as Content Delivery Networks (CDNs) like Akami typically cache static pages at end nodes and deliver them to customers from physically closer nodes, reducing overall latency. GlobeDB [12] automatically replicates Web application data. It does not replicate web pages, but rather the underlying data in the web application database allowing for a Web application to be rendered at end nodes. Distributed object caches such as MTCache, DBCache, and Memcached, are key-value stores that can also store the results of database queries as the value of the query key and allow for data replication for Web applications.

C. Connectivity

A key requirement for this architecture to be possible is pervasive connectivity and *reachability*. It is not enough for the device to have internet connectivity, but it must also be reachable from the internet. This is a key challenge for our project, since almost all cell phone providers and WiFi access points hide their clients behind network address translation (NAT) and or a firewall which blocks reachability from the internet. A lot of work has already been done on ensuring reachability, with the key technologies being IPv6, which includes Mobile IP, [13] as well as the Unmanaged Internet Architecture (UIA) project [14], [15].

One of the primary motivations for using NAT is the scarcity of IPv4 addresses, as it allows multiple devices to connect to the internet through one public IP address. IPv6 solves this limitation, as it has 7.9×10^{28} times as many addresses as IPv4. Additionally it includes Mobile IP, which allows devices to be located and contacted by their "'home address"' regardless of where the are on the internet.

Another very relevant project is Unmanaged Internet Architecture (UIA), as it seemingly solves the connectivity and reachability issue even in the current state of the internet, however it requires some assistance from devices with public IPs to enable NAT traversal. Additionally it provides other very relevant services such as security, group management, overlay routing, and personal naming.

In short, while connectivity is an important part of our project, it seems that it is largely a solved problem, and thus will not be our focus. While it is unlikely cellular provides will enable public reachability to cell phones, this is due to business and security reasons rather than a technological limitation.

D. Security

A key concern when decentralizing the Internet is the storage and distribution of data among peers and friends. Trusting other devices with our personal data causes uncertainty, specially regarding integrity and privacy of the data. There exist vast previous work in data security, however we would to focus in secured distributed systems and data storage. In distributed systems, there is no central system to protect, thus all nodes must collaborate to ensure security [16]. A common approach of these systems is the peer-to-peer distributed hash lookup systems [17], [18], [19], in which lookups for keys are performed by series of nodes and forwarded to the node ultimately responsible for the key [16]. Overall, approaches to ensure confidentiality, integrity and availability (CIA) must have capabilities of encryption, stringent access controls to data, backup and safe storage of data [20].

E. Peer-Peer

A common method to activate peer-peer communication, is the utilization of hashmaps. In our project peer to peer communication is based on a secure cluster which includes our trusted devices, thus a hashmap only includes the reachable and trustful devices. Existing hashmaps implementation works as follows. Every time the device detects a new reachable device, it will put this divide into its own hashmap. And every time a device receives a message, it will look up the destination in its hashmap, if the destination is not found, it will make a connection with the destination. If it cannot, it will flood the message until get a signal of success. If it cannot receive a success signal for a period of time, maybe be 5 milliseconds, it will send the signal to the central server. In contrast to this approach, our system will not depend in a central server. There

exist a plethora of current P2P frameworks such as Chord, Pastry, PAST, Tapestry, and Viceroy [21].

III. Methodology

The realization of our architecture can be divided in to two distinct stages: First we will leverage existing technologies to establish the framework for our system, in particular connectivity, reachability, and basic services. Second, we will build new technologies and applications to provide the key functionality of our decentralized system.

A. Framework

The first step in realizing our architecture is establishing connectivity and reachability to our devices. As discussed in Section II-C, existing work has already addressed, and largely solved, this issue, thus it will not be our focus. Unfortunately these methods have not been widely deployed, are blocked, or have only been tested in lab settings. To bypass this challenge, we intend to use a VPN server, such as OpenVPN [22], running on an RECG lab server to allocate static public IPs to our devices. We hope that this will enable our devices to be publicly accessible, as well as provide a reliable way for them to communicate with each other. However, if this approach fails, we may have to fall back to other methods such as UIA [14], [15], or limit the scope of our experiments.

The next component in our architecture is device naming and management. Conventional DNS provides human friendly names as well as domain management, and, in more advanced systems can also provide load-balancing and failover. In our architecture we will additionally use it help specify our personal "'mini-clusters"' and failover order. For example, each group member will have their own sub-domain, then further sub-domains will be used to manage that users' devices and cluster. Figure 3 is an example of a possible namespace. More specifically we intend to use a name such as "'phone.adriana.recg.rice.edu"' or "'laptop.adriana.recg.rice.edu"' to point to Adriana's phone and laptop respectively, then perhaps "'failover1.adriana.recg.rice.edu"' up to "'failoverN.adriana.rice.edu"' to specify the N failover devices in Adriana's cluster. Similar naming would be used for each persons' cluster. Notably these names would only be used internally, and would not be seen by the users. While DNS is a centralized service, thus breaking our decentralized mantra, we use it out of convenience and time considerations; other decentralized naming systems could also be used similarly, such as the one proposed in UIA.

The final "'off-the-shelf"' component of our framework is open-source server software, such as postfix, sendmail, apache, etc., which will provide the basis of our communication services. Combined with the reachability and naming described above, this provides a complete framework for our decentralized framework, which should provide full communication and service, which are even backwards compatible with the existing email and web services. However, this framework lacks two key features: redundancy and failover. Additionally, this framework provides some unique opportunities for optimization, which we elaborate on below.

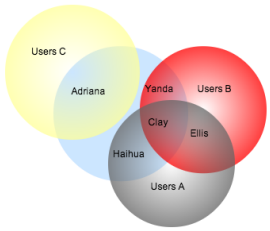

Fig. 3. Example Namespace.

B. Performance Metrics and Analysis

Once we have built our decentralized architecture, it is critical to analyze its performance. We will analyze:

- Uptime, Reachability, and Failover
- Failure Resistance and Recovery
- Load balancing and power efficiency

C. Key Issues

Even when building a decentralized architecture, we are still required to provide reliability by having 100% availability of the data and fast accessibility. In order to achieve the goals of reliability, 100% availability and fast accessibility, our system must overcome the following.

Failover. In a decentralized architecture, the number of point of failures are increased. In a centralized architecture, if a central server fails, it is easily trackable and detectable. However, in the case of a distributed architecture, the points of failure are multiplied. Even though, multiple point of failures is helpful in security purposes, it makes the system more prone to errors because detecting the failure is harder to then activate failover mechanism. In order to address this issue, our system relies on data replication or redundancy. Data will be distributed across multiple nodes, where we will provide a redundancy level that is proportional to the size of the network and load.

Furthermore, we identify there exist varying requirements in failover strategies for different applications. In the case of real-time applications, such as chat or VoIP, if the user is unavailable or offline, there is no necessity in activating failover to other people's devices. Instead, failover for real-time applications should occur between the devices of the same owner. For example, if my phone is unavialble, my calls can failover to any other of my active devices.

In general, by storying multiple copies of the data our system will combat failovers. Moreover, our system will consider application base failover possibilities. Furthermore, as explained bellow, our system will take into consideration device capabilities for data distribution, thus reducing the possibility of failovers from power outage or other device dependent issues.

Diversity in Device Capability. Since, data will be store in various types of devices, our system is required to be robust to the different device capabilities. A key example of diverse capability is power or battery life. Some devices might be able to have longer up-time than others, thus our system is required

to adapt to such differences. A possible approach is power adaptation, by analyzing remaining battery life of a device, to then perform decisions of reducing transmission power or even remove such device from the cluster. Another example of diverse capability is the storage capability of a device. Some devices might be capable of storing larger amounts of data than others. Overall, our system will approach different device capabilities by analyzing each device, building a feedback infrastructure and performing decisions based on each device capability. Through this we ensure data distribution is performed properly in order to achieve our goals.

Mode of Connectivity. Different modes of connectivity will be supported and tested for different devices. For instance, a mobile phone could connect to the framework using a Wi-Fi connection direct to local network services. Alternatively, it could connect using a VPN through a cellular telephone network. Ideally, the device should make a connection that supports the best overall user experience for the device user as well as the users consuming services provided by the mobile device. Bluetooth would be another possible connection mode, but is probably beyond the scope of this project. Laptop devices could connect in a wired or wireless fashion to the framework. Each connectivity method would consume a different set of resources and provide a different level of service. In our project we would like to not only make the method of connectivity as easy as possible for the end user, but will also measure the performance tradeoffs of connectivity modes.

IV. Expected Outcomes and Schedule

A. Expected Outcomes

The expected outcomes of our project can be divided into four stages. In stage 1, we plan to identify, install and deploy our framework with off-the-shelf technology (VPN server with public IPs and DNS) to meet our basic needs. Next, in stage two we plan to extend our framework beyond off-the-shelf technologies in order to ensure the framework is capable of executing all our needs in terms of analysis and system features. Third, we will integrate use cases to our framework, meaning we will deploy applications that will allow us to test our system. Example applications we consider in deploying are wordpress or open source source social media. In the last stage, we plan to test and analyze performance of the system in terms of failover, power and other device capability dependent issues.

B. Schedule

- Beginning March - Framework Setup: Get connectivity and reachability on phones (VPN server with public IPs and DNS)
- Early March - Framework Setup: Servers installation (Web and Email)
- Mid March - Midterm Presentation: We will have basic experimental framework installed so we can start working on novel issues (replication, failover, and power).
- Ending March - Data Replictaion: Basic replication scheme working
- Early April - Failover: Basic failover mechanism developed and implemented
- Mid April - Power: Experiment with performance and power
- April 22- Final Report: Integrate results into the final report
- April 22 - Final Presentation: Polish and integration into "app"

References

[1] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM, 2007, pp. 165–178.

[2] D. Schuff and R. St. Louis, "Centralization vs. decentralization of application software," *Commun. ACM*, vol. 44, no. 6, pp. 88–94, Jun. 2001. [Online]. Available: http://doi.acm.org/10.1145/376134.376177

[3] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user-defined privacy," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 135–146, Aug. 2009. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592585

[4] T. Xu, Y. Chen, X. Fu, and P. Hui, "Twittering by cuckoo: Decentralized and socio-aware online microblogging services," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 473–474. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851270

[5] F. Xhafa, V. Kolici, A.-D. Potlog, E. Spaho, L. Barolli, and M. Takizawa, "Data replication in p2p collaborative systems," in *Proceedings of the 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, ser. 3PGCIC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 49–57. [Online]. Available: http://dx.doi.org/10.1109/3PGCIC.2012.26

[6] H. Shen and Y. Zhu, "A proactive low-overhead file replication scheme for structured p2p content delivery networks," *J. Parallel Distrib. Comput.*, vol. 69, no. 5, pp. 429–440, May 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2009.02.008

[7] H. Shen, "Ead: An efficient and adaptive decentralized file replication algorithm in p2p file sharing systems," in *Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, ser. P2P '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 99–108. [Online]. Available: http://dx.doi.org/10.1109/P2P.2008.37

[8] M. Vardhan, A. Goel, A. Verma, and D. S. Kushwaha, "Demand based file replication and consistency mechanism," in *Proceedings of the 2012 Third International Conference on Computer and Communication Technology*, ser. ICCCT '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 335–339. [Online]. Available: http://dx.doi.org/10.1109/ICCCT.2012.74

[9] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 188–201, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502053

[10] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 202–215, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502054

[11] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003. [Online]. Available: http://dx.doi.org/10.1109/TNET.2002.808407

[12] S. Sivasubramanian, G. Alonso, G. Pierre, and M. van Steen, "Globedb: Autonomic data replication for web applications," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. New York, NY, USA: ACM, 2005, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/1060745.1060756

[13] S. E. Deering, "Internet protocol, version 6 (ipv6) specification," 1998.

[14] B. A. Ford, "Uia: a global connectivity architecture for mobile personal devices," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[15] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris, "Persistent personal names for globally connected mobile devices," in *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006, pp. 233–248.

[16] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Peer-to-Peer Systems*. Springer, 2002, pp. 261–269.
[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network*. ACM, 2001, vol. 31, no. 4.
[18] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Springer, 2001, pp. 329–350.
[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4. ACM, 2001, pp. 149–160.
[20] L. M. Kaufman, "Data security in the world of cloud computing," *Security & Privacy, IEEE*, vol. 7, no. 4, pp. 61–64, 2009.
[21] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and Tutorials*, vol. 7, no. 1-4, pp. 72–93, 2005.
[22] M. Feilner, *OpenVPN: Building and integrating virtual private networks*. Packt Publishing Ltd, 2006.