An Experimental Evaluation of Delay-Tolerant Networking with Serval

Lars Baumgärtner*, Paul Gardner-Stephen†, Pablo Graubner*, Jeremy Lakeman†, Jonas Höchst*, Patrick Lampe*, Nils Schmidt*, Stefan Schulz*, Artur Sterz*, and Bernd Freisleben*

*Department of Mathematics & Computer Science, University of Marburg, Germany
Email: {lbaumgaertner, graubner, hoechst, lampep, schmidt89, schulzs, sterz, freisleb}@informatik.uni-marburg.de
†School of Computer Science, Engineering & Mathematics, Flinders University, Australia
Email: {paul.gardner-stephen, jeremy.lakeman}@flinders.edu.au

***Abstract*—Serval is an open-source, delay-tolerant wireless ad-hoc networking system designed to allow communications anywhere and anytime, despite the total loss of supporting telecommunications infrastructures provided by mobile phone operators. In emergency situations, Serval can be used to establish a disaster-response communications network spontaneously formed by mobile phones and/or battery powered wireless routers. In this paper, we present an in-depth experimental evaluation of Serval for various network setups and usage patterns, including simulated long term use. The focus of our evaluation is on the delay-tolerant aspects of Serval, providing insights into the scenarios where Serval can be deployed with satisfactory quality and performance characteristics. Furthermore, since mobile phones have a limited battery capacity, we take a closer look at the battery drain resulting from using Serval over different communication links, such as WiFi and Bluetooth. Our purpose in providing these analyses is to understand the current capability of Serval and identify any areas where further improvement is required, and to provide a summary of current readiness of Serval in advance of planned pilots in the Pacific region.**

I. Introduction

The unfortunate reality is that each year disasters and emergencies in many places around the world happen, and that a common feature of these events is that partial or complete loss of communications capacity occurs. Even without the loss of communications capacity in a disaster, there are significant challenges to providing effective information for those affected [1]. The loss of means of communications serves to compound the difficulties and sufferings faced by those affected [2]. There is, therefore, a moral imperative to seek out means of finding ways to restore, or better, sustain communications during and following such adverse events.

The Serval Project [3], [4], [5], [6] is one of a number of endeavours that seeks to respond to this moral imperative. Its objective is to allow people to use mobile telephone handsets to communicate anywhere, anytime [5]. The project seeks to achieve this by creating protocols, writing software, including mobile apps, and creating complementary hardware devices that, together, are able to replicate many of the functions of a conventional cellular network to some degree. The goal is not the replacement of cellular networks, but rather provisioning the best possible set of functionality and quality of service that is feasible, without requiring any conventional infrastructure.

Currently, pilots are being planned in the Pacific region and in Outback Australia over the coming months. The pacific trials that are sponsored by the Pacific Humanitarian Challenge[1] in particular will involve the provision of Serval technology to the general public in regions that are particularly vulnerable to natural disasters. It is therefore imperative that the behaviour of the technology be sufficiently characterised, so that informed decisions can be made, and where any current deficiencies might exist, that they can be identified, and thus be scheduled for remediation. There is a similar need for the activities of the NICER[2] project in Germany. NICER is exploring how infrastructureless information and communications technology can establish links between people in the event of a crisis, thus enabling them to work together to overcome the crisis.

In this paper, we present an in-depth experimental evaluation of the delay-tolerant networking (DTN) aspects of the Serval software stack for various network setups and usage patterns, including simulated long term use. The evaluation is based on a simulation and emulation environment to provide insights into the scenarios where Serval can be deployed with satisfactory quality and performance characteristics, without requiring the expense and complication of deploying large and potentially costly physical test networks. Since battery capacity is limited on mobile phones, we take a closer at the battery drain from using Serval over various communication links, such as WiFi and Bluetooth. The contributions of this paper are:

- A hybrid simulation and emulation environment is presented that allows us to run real OpenWRT[3] firmware images in an emulator, in contrast to mere simulations where only the DTN protocol can be tested.
- Various network topologies, ranging from many 1-hop neighbours and a 64-hop chain to more realistic merging islands connection schemes are evaluated.
- Several test cases mimicking common functionality, such as file distribution, messaging and peer discovery, and typical user behaviour, such as rapid bulk insertion of

[1] http://pacifichumanitarianchallenge.org/
[2] https://nicer.network/en
[3] https://openwrt.org/

content, writing periodic text messages, and adding different types of content every now and then, are considered.
- Different file sizes are examined to reflect different patterns of mobile phone usage, such as sharing text files (GPX data, ebooks, messages), images (map tiles, pictures), voice and video recordings (eye-witness video footage, voice memos, diaries).
- All test data, scripts and topologies are freely available and can be adapted to test other software[4].

The paper is organized as follows. Section II discusses related work. Section III presents the basic concepts of Serval. The experimental setup is described in Section IV, and experimental results are presented in Section V. Section VI concludes the paper and outlines areas for future research.

II. Related Work

There exists a wide range of related work addressing emergency communications needs and solutions, beyond what is possible to cover in this paper [7]. Nonetheless, many of the solutions in this space can be classified according to (1) the communications medium/media and modulation(s); and (2) the architectural model(s) used by each solution.

Communications media include WiFi, Bluetooth, WiMAX, GSM, TETRA digital radio, and various analog two-way and digital microwave, UHF, VHF and HF radio systems, as well as wired analog or digital systems, and satellite based systems, all available from various commercial vendors.

The architectural models can be often classified as either infrastructure-oriented, distributed (including peer-to-peer ad-hoc systems), or hybrid architectures of both approaches.

Several systems support multiple transport modalities. For example, WISECOM [8] is an infrastructure-oriented system that seeks to provide a comprehensive approach to post-disaster communications, using satellite for global connectivity and a wide range of media and modulations. A significant challenge with such systems is their overall complexity, and their dependence on a sophisticated Internet-side infrastructure.

Distinct from the transport media, considerable work has been done on designing network protocols and frameworks for emergency communications using various selections of the media and modulations listed above [9], [10], [11]. A resulting problem in this diversity is that interoperability can be a signifcant challenge and requires ongoing effort to contain and improve this situation [12], [13].

Mobile applications are also becoming more prominent in the emergency communications space [3], due to the increasing capability of modern smartphones. Several systems also employ DTN principles to mitigate the challenges that arise when forming networks from end-user devices, and without adequate supporting infrastructure [14]. Such systems are particularly relevant, due of their ability to operate when faced with the failure of infrastructure, which is a common feature in disasters and emergencies [7].

For example, FireChat[5] is a DTN system for sending messages, but it lacks openness. Other DTN systems such as SPAN [15] and Briar[6] only support specific target operating systems such as Android, and SPAN does not provide applications built on top of it. Furthermore, Forban[7] can spread files opportunistically in a DTN manner, but lacks protocol support for direct private file transfers, messaging or routing.

Liu et al. [16] have developed a DTN based mobile microblogging app for censorship resistant communication. Their focus is on the app's energy consumption in an 802.11 ad-hoc network, ignoring other means of communication such as Bluetooth or WiFi in AP mode and limiting the system to specific rooted Android devices in ad-hoc networking mode. Also, there is no support for sending large files, such as videos.

Ntareme et al. [17] have presented an approach based on Android phones using a store-and-forward architecture. Services such as email are transparently delivered via DTN, but the solution requires special server software in addition to the Android app. Energy and bandwidth consumption were measured, but scalability and performance in different scenarios were not evaluated.

Heimerl et al. [18] attempt to solve the problem of poor cellular coverage and power outages in rural areas by using low-cost GSM hardware and a system for reduced power consumption. While this approach is interesting for feature phones and services such as voice calls and text messages, it still requires infrastructure to function.

III. Serval

The basic concepts of Serval are presented below.

A. Overview

Serval is centered around a suite of protocols and technologies designed to allow ad-hoc infrastructure-independent communications [3], [4], as illustrated in Fig. 1. The goal is to provide infrastructure-independent versions of many of the services that are commonly used on smartphones in conjunction with the Internet and/or cellular networks, e.g., voice calls, short text messaging (SMS), voice mail, social media, as well as file and image transfer.

The Serval Mesh protocols purposely take a contrasting approach to that of using IP (v4 or v6) as the basis for forming mobile ad-hoc communications networks (MANETs) [5]. The reason for this is that despite billions of dollars of research and development work, IP-based MANETs still struggle, and face a number of significant challenges that limit their real-world use, e.g., address allocation, the need to maintain a routing table, authenticity and integrity of communications, and the need for relatively reliable and stable end-to-end connectivity for such systems. Instead, Serval uses 256-bit public cryptographic keys as the primary network identifier, the so-called Serval ID (SID), and also includes a rich security model that facilitates confidentiality, integrity and authenticity by design, and does not require a Trusted Third Party (TTP) to operate. It also includes a store-and-forward DTN protocol (Rhizome), allowing network operation in the absence of end-to-end connectivity.

[4]https://github.com/umr-ds/

[5]http://opengarden.com/firechat

[6]https://briarproject.org/

[7]http://www.foo.be/forban/

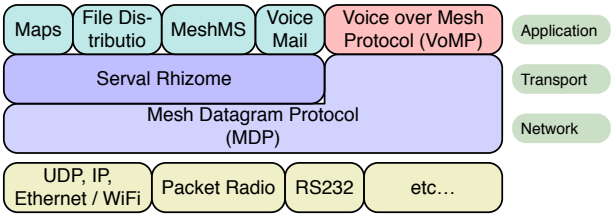


Fig. 1: The Serval technology stack

B. Rhizome and Delay Tolerant Networking

Rhizome is a simple bundle protocol that principally defines data units as *bundles*, consisting of an optional payload, together with a *manifest* that contains necessary meta-data. *Manifests* have a hard size limit of 1 KB to improve efficiency, and must also contain a cryptographic public key that is used to protect the integrity and authenticity of the manifest itself. The manifest may also contain a cryptographic hash, indicating that it has an associated payload, together with other meta-data, such as mime-type, Rhizome service tag, file-name, and SID of the sender and/or recipient, as appropriate.

While the Rhizome implementation includes several transports for Rhizome, including HTTP, packet radio and the Serval MDP protocol described below, the protocol is purposely agnostic of the transport, to allow other transports to be added. The intention of this is that any transport that is capable of carrying bytes of data can be used to transport Rhizome data.

As a simple state-less flooding protocol, Rhizome requires no routing table, and never requires that two parties have an end-to-end connection for them to communicate. That is, the Rhizome protocol is always focused on single-hop communications, with multi-hop communications emerging as a natural consequence of *bundles* replicating among nodes.

Rhizome is used as the basis for the SMS-like Mesh Messaging Service (MeshMS) [6], and file distribution, including software updates. It is also planned to implement a twitter-like micro-blogging service using Rhizome.

C. MDP, MSP and Node Discovery

In addition to the Rhizome DTN protocol, Serval also includes a real-time packet-switched protocol, the Mesh Datagram Protocol (MDP) that is generally similar to UDP/IP, but uses SIDs instead of IP addresses, and includes encryption, authentication and integrity features by default. The TCP-like Mesh Streaming Protocol (MSP) is layered atop MDP to provide reliable data streaming. Various services can be implemented atop MDP and MSP, including the VoIP-like Voice over MDP Protocol (VoMP).

MDP routing uses an OSLR- and BATMAN-inspired [19], [20] ad-hoc protocol for both node discovery and maintaining a routing table, that facilitates multi-hop routing of packets. In order to reduce packet sizes, address abbreviation is used, so that only the minimum number of bytes of a SID is required to uniquely identify a node among its direct, i.e., 1-hop neighbours. This reduces the header size in the common case to be smaller than that used for IPv6.

IV. Experimental Setup

The experimental setup for our in-depth evaluation of Serval, including the hard-/software environment used, the parameters measured, the network topologies chosen, and communication scenarios, is presented below.

A. Simulation/Emulation Environment

To evaluate the performance in a realistic manner, we have developed a simulation/emulation system called *MiniWorld* based on QEMU/kvm and Linux networking code to achieve full system emulation. This gives us the opportunity to use the OpenWRT build chain for building router images that include Serval. OpenWRT is also used on real world routers such as TP-Link MR3020 or the Mesh Extenders of the Serval Project. Having a full operating system with its own network stack running on each node gives a much better picture of real life performance than pure protocol simulation.

All tests are performed on a 64 core AMD Opteron 6376 CPU with 256 GB RAM, simulating up to 100 virtual nodes, each one with 512 MB RAM and 2 GB of storage space. These quite limited values allow us to investigate how Serval performs on older smartphones like the original Samsung Galaxy S or similar, which are common in developing countries.

B. Measurements

Standard Unix tools are used to measure system properties, with a measuring interval of one second. For memory consumption, CPU and I/O usage *pidstat*[8] is used to monitor the statistics of the Serval process from within a node. Disk space is measured with *du* and *df*, both from the GNU coreutils[9]. Network usage is measured on the *MiniWorld* bridge interfaces of the host system using a custom Python tool[10] based on *libpcap*[11]. Insertion points in time for the Rhizome store are derived directly from Serval's log, while the general file count is logged using direct *servald* calls.

C. Network Topologies

Several network topologies are studied, as shown in Table I. The *Hub* topology connects 48 nodes with each other. It represents a scenario with a high number of direct neighbours all using bandwidth, flooding each other with status information and new files, sharing the same transport channel. Typically, the number of direct neighbours is limited by the radio range of WiFi or Bluetooth (i.e., often less than 48). Thus, *Hub* is challenging for Serval and also the radio link itself.

The *Chained* topology consists of a chain of 64 nodes, thus the last node is 63 hops away from the first node. Typically, network connections over the Internet require less than 16 hops. In a delay-tolerant mobile mesh network, more hops might be needed for messages to reach their destination compared to static networks physically optimized for minimum hop numbers and maximum throughput.

[8] http://sebastien.godard.pagesperso-orange.fr
[9] http://www.gnu.org/s/coreutils/
[10] https://github.com/umr-ds/serval-tests/blob/master/netmon.py
[11] http://www.tcpdump.org

TABLE I: Topologies

Name	# Nodes	Description
Hub	48	All nodes connected to each other
Chained	64	Pair-wise connected
Islands	100	Partitioned islands, merging over time

The *Islands* topology represents a partitioned network that slowly merges over time. At the beginning, there are 100 nodes in small islands with only a few neighbours. Between these small islands there are no links, but after a predefined time a few of them merge together, exchanging all their information that they have collected so far. Finally, there are two big islands where one node acts as a bridge between the two, and all accumulated data from one island has to pass through this node to propagate to the other island.

All topologies are used in two configurations, one modeled after the common 802.11g standard with a 54 Mbit/s limit on each link and one with no bandwidth limitations.

D. Scenario Tests

Based on these topologies, we designed several tests, as shown in Table II.

Idle (I) simply starts Serval and waits until all nodes have found each other. This test serves to evaluate how long the discovery phase takes in various network setups and how much traffic Serval produces while idling.

Mass Files (MF) pre-generates a number of files and inserts them at one specific node. The goal is to evaluate whether Serval can handle a large number of files at once. Propagation through the network is observed to reveal problems related to high bandwidth, storage and/or CPU usage.

Mass Messages (MM) is designed to test the messaging subsystem of Serval by flooding the network with text messages. A number of messages is sent at once to every single node in the network no matter if it is currently reachable or not.

Periodic Files (PF) is designed to observe the long-term behaviour of the system. Files are added at random points in time by every node. A real world analogy is: people taking pictures occasionally and sharing them with everybody else.

Periodic Private Files (PPF) is a special case of *PF* where files are not shared with the public but sent to a randomly chosen recipient.

Periodic Messages (PM) is designed to evaluate the Serval messaging subsystem. These messages are also directed to a specific recipient and are not meant for the public.

Combined (C) is designed to run all periodic tests (*PF, PPF, PM*) at once. Similar to real life situations, the nodes change their behaviour and there is a competition for the resources in the network. Broadcasting files, sending files to "friends" and writing text messages all have different requirements.

E. Data Sent

Text messages consist of a fixed string plus a timestamp in milliseconds when a message was sent. Since these messages are meant to mimic real world chat, the total string length is kept small (53 characters). According to a chat study of Battestini et al. [21], text messages sent by males had an average length of 47 characters and for females 58 characters.

TABLE II: Scenario Tests

Name	Short	Description
Idle	I	Node discovery, no actions triggered
Mass Files	MF	Insert bulk of file set at once
Mass Messages	MM	Insert bulk of messages at once
Periodic Files	PF	Periodic adding of files
Periodic Private Files	PPF	Periodic adding of private files
Periodic Messages	PM	Periodic sending of messages
Combined	C	All periodic tests together

TABLE III: Test File Sets

Name	Sizes	Description
Small	64K, 256K, 512K	Small pictures, map data, text files
Medium	1M, 5M, 10M	Camera pictures, audio recordings
Large	25M, 50M, 100M	Recorded video
Mixed	all of the above	-

Files have different file sizes representing different types of data, as shown in Table III. The *Small* file set contains randomly generated files ranging from 64 KB to 512 KB; large text files, ebooks, small pictures or other data such as map tiles typically have these sizes. In the *Medium* file set we have files between 1 MB and 10 MB, which is nowadays the size of pictures taken with mobile phones or some audio recordings. Recorded video or software bundles are represented in the *Large* file set and are generated in the range from 25 MB to 100 MB. Finally, there is a *Mixed* file set where small, medium and large files are included.

F. Test Execution

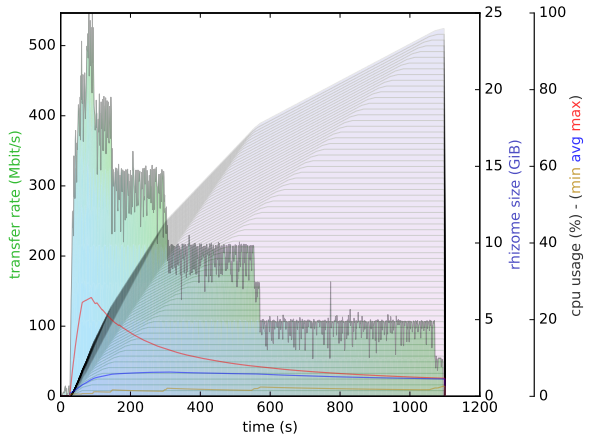
All file related tests were performed with all four file sets, every test was executed on all topologies with limited and unlimited bandwidth resulting in a total of 114 tests. While some tests (e.g., *MF*) are count-based and terminate after every node has received a specific number of files, other tests (e.g., *PPF*) are time-based - always running for the same duration. We performed 5 iterations of each test, resulting in a total of 570 test runs.

V. Experimental Results

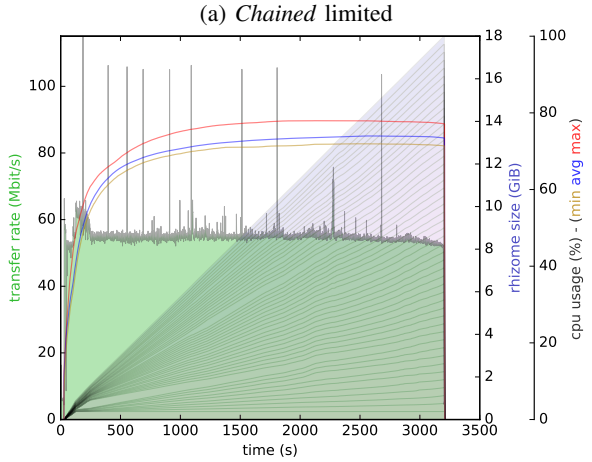
In this section, various results regarding Serval's behaviour during the experiments are presented.

A. Idle Behaviour

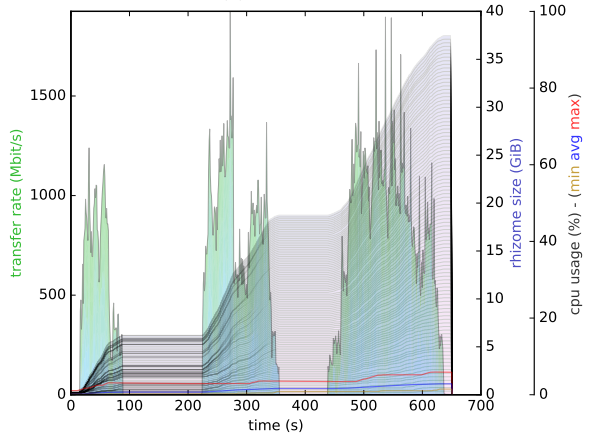
To investigate the idle behaviour of Serval, we looked at network traffic, CPU load and memory usage after the initial discovery phase, without triggering further actions. In every scenario, whenever Serval is started, there are peaks in the network load, in the *Chained* and *Hub* topologies at approximately 10 to 12 Mbit/s. After this peak, *Chained* has a summed average network traffic of around 0.7 Mbit/s, whereas the nodes in *Hub* produce 6 Mbit/s. This behaviour is



(a) *Chained* limited



(b) *Hub* limited



(c) *Islands* limited

Fig. 2: *MF Mixed*: Rhizome store size, network and CPU load.

caused by Serval's information distribution strategy, because it announces status information, such as the list of files in Rhizome, periodically via broadcasts. Since there are 47 neighbours for each node, traffic is relatively high in the *Hub* topology. *Islands* has extrema whenever partitions merge. The traffic during peaks grows with the number of nodes.

CPU usage of the Serval process correlates with network load in our scenarios, but never gets larger than two percent per node. Serval uses around 4 MB of memory in all scenarios.

Moreover, the discovery time of each topology is different. For *Hub*, the average time of a full network discovery is approximately 5 seconds, since every node has a direct connection to all others. In contrast, the *Chained* topology takes about 20 seconds, because announcements have to be forwarded through all other nodes.

In some experiments, Serval's address abbreviation (Sec. III-C) mechanism caused conflicts under special circumstances, depending on the keys and when different nodes announce themselves for the first time. If a node already has seen another node with the same abbreviated address, it is ignored, potentially causing a partitioning of the network. To circumvent such effects, we modified Serval to generate unique prefixes for the desired node number in our tests.

B. Hub Constraints

For *Hub*, a single bridge interface was used to connect all nodes. Since each node is a single hop away from all other nodes and Serval uses broadcast packets to announce meta-data (e.g., the files of a node), each node is flooding all neighbours with this information. Since the number of adjacent nodes affect the CPU consumption of the respective node, in the *Hub* topology the CPU usage is always higher than in the corresponding test in *Chained* or *Islands*, due to the high number of direct neighbours.

C. Topology Characteristics

Fig. 2 shows *Mass Files* tests with a *Mixed* file set in different topologies. It shows how transfer rate in Mbit/s, size of the Rhizome database and the CPU usage change over time. The transfer rate is stacked for all links. The Rhizome size is the stacked database sizes of all nodes.

Fig. 2a shows a limited (802.11g) *Chained* topology, in which five phases are visible, caused by the Rhizome prioritization based on file sizes. Small files are delivered first and therefore can be distributed earlier by the following nodes. The bigger the files get, the less total network utilization is achieved. Despite this effect, a constant stable data flow is visible, and the Rhizome store grows constantly. The maximum CPU load correlates with network usage, since the most active network nodes do have the highest CPU usage.

In Fig. 2b, a limited *Hub* topology is shown. Though a constant 54 Mbit/s data flow is visible, the spikes exceeding 54 Mbit/s are measurement errors, caused by differing network backend and traffic monitoring timers. With a constant network load caused by the file transfers, the disk usage also grows linearly as expected in this case, meaning that the network load is not dominated by status and management information but real content distribution. Compared to Fig. 2a the average CPU usage is about 10 times higher, as explained in Sec. V-B.

For *Islands*, CPU usage increases every time the network changes. Looking at *Periodic Files* tests, the max. CPU load rises to 15% when large files are inserted, since they have to be redistributed among the other nodes. Fig. 2c shows the *Mixed* file set in *MF*, which peaks at around 7% CPU load. Since many of the files already exist on various nodes, every time new network connections are set up, the impact on the

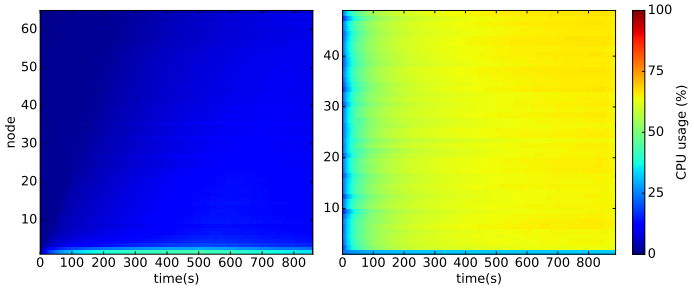


Fig. 3: *MM* CPU usage over time. Left: unlimited *Chained*, right: unlimited *Hub*.

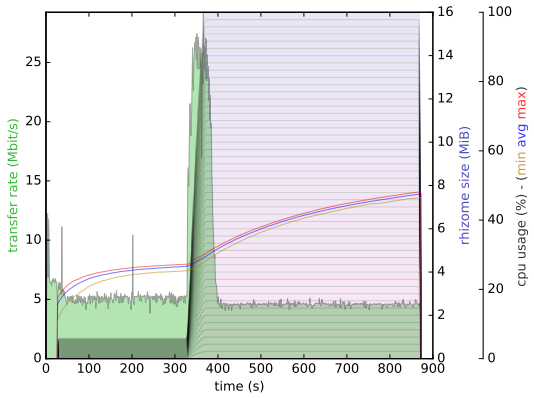


Fig. 4: *Hub* limited *PM*: Rhizome store size, network & CPU

CPU is relatively low compared to *Hub*. In general, smaller files have a negligible impact on the CPU.

The *Periodic File* tests with small sizes do not show any unexpected behaviour in terms of CPU consumption in *Chained*, the CPU peaks at about 10%. When the files are encrypted as in *PPF*, the CPU utilization is slightly higher, at about 15%, due to CPU intensive cryptographic operations.

The file size influences CPU utilization, which greatly impacts the inserting node. For instance, when sending *Small* files in *Chained*, there is no significant change of CPU utilization compared to idling, whereas file set *Large* utilizes the CPU up to 35%. Bigger files lead to more time consuming hashing, as it is required by the corresponding protocol. Thus, every node receiving the file needs to compute a hash, verify and redistribute it, which also leads to a higher load.

In terms of CPU usage, *Islands* is a combination of *Chained* and *Hub*. CPU usage does not exceed 50%, since the total number of neighbours per node is not as high as in *Hub*.

In the message based tests, the measured CPU consumption correlates with the number of messages sent. For *MM*, the behaviour differs depending on the topology used. Fig. 3 shows the CPU usage per node of two experiments over time. Using *Chained*, the inserting node peaks at 30% CPU load compared to the receiving nodes, which consume about 15%. Using *Hub*, the load of the inserting node remains the same. In contrast, the receiving nodes constantly consume about 65% CPU. *Hub* suffers from the broadcast overhead (Section V-B), but this does not fully explain the high load, as the sending node is not affected. Further investigating this behaviour, we tracked it back to recurring hashing and encryption in Rhizome Journal syncing, which is the core of MeshMS messaging.

PM results differ from *MM*. For *Chained*, the CPU utilization is relatively low at about 15% maximum. This correlates with the CPU load of the non-inserting nodes in *MM*. Since they are added periodically, the CPU overhead is negligible here. *Hub* behaves differently than in the file based tests or *MM*: The *PF* tests show that in every topology the more files are injected in the network, the more CPU is needed to handle the broadcast packets. Messages are not announced further after reaching their destination and being acknowledged by the recipient. The obvious consequence should be that the CPU usage decreases. However, as indicated by Fig. 4, once the CPU peaks at about 25%, it does not settle any more, but increases even further, although the network load decreases to the idle level and the Rhizome database size is at its maximum, which indicates that all messages have arrived. This behaviour cannot be transferred to *Islands*, where the inserting nodes peak at about 40% and all other nodes do not exceed 15%.

For *C* tests, the general CPU usage is similar to other file based tests. The only difference is the fact that in *Chained* and *Hub* the CPU usage increases by 5% after about 500 seconds and also correlates with the network load, similar to the behaviour depicted in Fig. 4. This problem emerges when sending messages over a longer time period. Since *Islands* is not in the final state at the beginning of the test in terms of the links between the nodes, this result can not be observed in this particular topology.

D. Network Performance

One goal was to test to what extent Serval is able to use available bandwidth. *Chained* was created to assess this.

The cumulative transfer rate using Rhizome in this topology reached 500 Mbit/s to 2 Gbit/s, depending on the file sets, with *Large* being the fastest. That is, up to 2 Gbit/s of traffic was being carried over the set of hops in the chain, with each seeing an average utilization of 32 Mbit/s. Tests that transfer large files over an unlimited network show that Serval is able to use even more bandwidth, since the highest measured transmission speed from one node to another can be up to 160 Mbit/s.

Using *Chained*, the hop-to-hop transmission time can be modeled, since node n is able to receive a file just after node $n-1$ received it. Fig. 5 shows the hop-to-hop transmission times of the *Medium* file set. The five files of each size are grouped into one box plot, while the colors present five different runs of each experiment. The median transmission times for 1, 5 and 10 MB files are 0.54, 1.06 and 1.85 seconds, and only 0.27 sec for 64 KB files. From these values, a simple correlation for the transmission time can be derived: $T(size_{MB}) = 0.16 \cdot size + 0.26$, which also holds for the *Large* set. The formula indicates a net transmission rate of around 31 Mbit/s, with a 0.26 sec delay.

The average speeds are lower, because files are exchanged node-by-node, and can only be spread to node $n+1$ after reaching node n, resulting in an effective end-to-end bandwidth, for

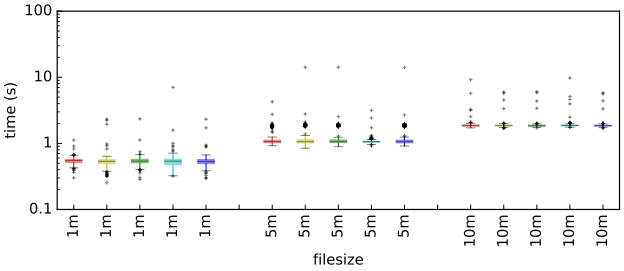


Fig. 5: *Chained* limited *Medium* file set: File-size-grouped hop-to-hop delivery periods of five runs.

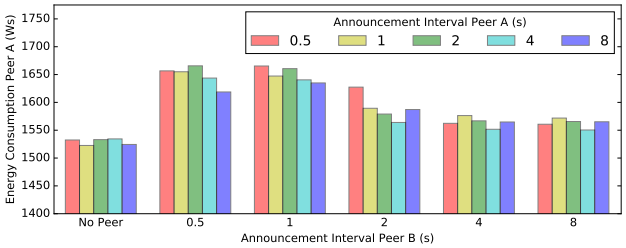


Fig. 6: Energy consumption of announcement intervals

a given bundle, inversely proportional to the number of hops. This compares favourably with end-to-end ad-hoc wireless routing protocols, where the effective end-to-end bandwidth drops by approximately half for each additional hop.

Briefly considering the different topologies, the network utilization in *Islands* for file based tests is generally about the same as in *Chained*, since each node has only a few neighbours, in contrast to *Hub*, which is always able to saturate all links due to the high degree of connection among nodes.

Messages in Serval are effectively transported as small files, with a payload size of 53 bytes in both *PM* and *MM* cases. The network load shows a behaviour similar to small files in the *PF* test, peaking at up to 40 Mbit/s at all topologies and regardless if the network is limited or not.

The network load for *C* tests in all topologies is similar to the file based tests, independent of bandwidth limitations. The only difference is the increase of the network load after about 500s on *Chained* and *Hub*, as shown in Section V-C.

In *Hub*, small files take between 1 and 4 min to arrive on the last node in the limited network links. This increases linearly, up to 20 min, with increasing file size. If the network is unlimited, transmission time reduces to between 18 sec and 9 min, depending on the file size. One difference between *Hub* and *Chained* is the runtime. *Small* files are transmitted faster in *Hub*, whereas *Large* files are faster in *Chained*. The time overhead for file announcements is relatively higher for *Small*. Even with a lower total bandwidth (*Hub*: 54 Mbit/s for 48 nodes vs. *Chained*: 54 Mbit/s pairwise), *Hub* can achieve faster transmission rates. The limitation of network speed does not influence this behaviour, only the overall transmission time increases.

The transfer times of messages differ from topology to topology. While it takes about 350 sec in *Chained* until all messages arrive at their destinations, it can take up to 900 sec in *Hub*. This again shows that the high number of 1-hop neighbours in *Hub* is challenging for Serval. The transmission time for messages in the *C* tests depends highly on the used file set, rather than on the topology and network speed. The reason is that the network is saturated with big files, which leads to overall higher transmission times for messages.

E. Energy Consumption

The *Idle* test in Section V-A showed network peaks caused by Rhizome status information announcements. Therefore, the energy consumption of the announcements is evaluated: Two devices send announcements in different intervals. Fig. 6 shows the energy consumption for peer A using different announcement intervals at peer A and peer B. With a 0.5 sec or 1 sec interval, the consumed energy is 9% higher than in idle state. With a 2 sec interval, the consumed energy is only 3% higher than in idle state. With a higher interval of 4 sec or 8 sec, only negligible decreases in energy can be achieved.

Furthermore, the power consumption during *MM* and *MF* tests were evaluated. Two peers were connected via an 802.11n WiFi Access Point. Peer A inserts files and messages into Rhizome in the same manner as *MM* and *MF* tests. The power consumption of peer B, a Raspberry Pi 3, is then measured with the Odroid Smart Power measurement device, an external power meter. The aim of these experiments is to measure the energy overhead for running Serval on a device, which allows conclusions about the power drain of Serval on battery-powered devices.

Fig. 7 shows the power consumption during different Rhizome file set insertions similar to *MF*. The file sizes are increased during the phases *f1-f4*. During *f1*, the file sizes are smaller than 1 MB, resulting in a negligible additional power consumption. The bigger the transmitted files are, the more power is consumed. The comparison between receiving files and sending files shows an unexpected behaviour: In all phases *f1-f4*, sending files is less expensive compared to receiving files, on the average between 0.05 and 0.1 W (3-6%). This counterintuitive result is caused by additional CPU consumption of the Rhizome checksum calculation during reception. Compared to a 1.53 W mean idle value of peer B, the power overhead introduced by Serval is between 0.01 and 0.13 W (1-8%) during phases *f1-f4*.

In another experiment, we measured the power consumption during different message insertions similar to the *Mass Messages* test. The results show a power consumption peak between 1.81 and 1.91 W during a short period of reception, followed by a phase of negligible additional power consumption. During the reception of 100 messages, a mean value of 1.69 W (10%) additional power consumption is measured.

A better energy efficiency during message transmission could be achieved by using Bluetooth. It consumes a significant amount of energy during device discovery, but has a lower power consumption during data transmission than WiFi. Due to the low energy efficiency (joule per bit) of Bluetooth compared to WiFi, it consumes significantly more energy for large data transmissions. During an experiment, we

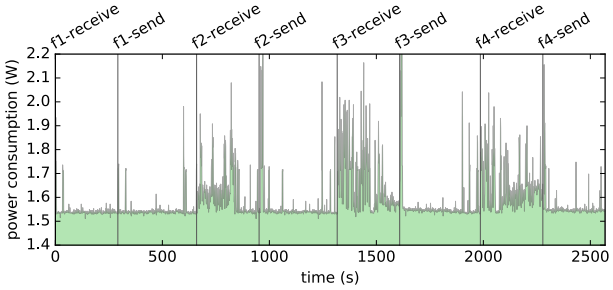


Fig. 7: Power consumption during different Rhizome file set insertions (*f1-f4*) similar to the *Mass Messages* test.

measured a 32 times better energy efficiency of WiFi compared to Bluetooth for files between 512 KB and 16 MB.

VI. Conclusion

In this paper, we have presented an in-depth experimental evaluation of the delay-tolerant aspects of Serval for various network setups and usage patterns. The results show satisfactory performance of Serval when deployed in partitioned scenarios and extreme examples of network topologies. Furthermore, we have analyzed Serval's energy consumption, having the limited battery capacity of mobile devices in mind.

In particular, our experiments indicate that there is a sweet-spot for the trade-off between up-to-dateness and energy consumption regarding announcement intervals. Furthermore, Serval can handle a realistic number of files over a longer time period. In the *Chained* topology, neither the CPU load nor the used network bandwidth leads to out of service situations. All tests with the *Hub* topology show that in a highly used network the announcements consume a considerable portion of the available bandwidth. In emergency situations or in long-term setups this could have a negative effect depending on the number of people in direct communication range. The *Combined* tests in our *Islands* topology demonstrate that Serval works flawlessly in adapting to heterogenous environments where users have different requirements at the same time and the topology changes over time.

There are several areas for future work. Mobility simulations should be carried out, preferably with real world movement patterns gathered from past events. More powerful hardware with higher numbers of nodes should be used to run the simulations and emulations, to further investigate Serval's scalability properties, particularly in highly-connected topologies, like *Hub*. The defect that has been exposed in the address abbreviation code should be rectified. An evaluation of Serval's non-DTN related features, such as voice calls, could further increase the attractivity of Serval as a solution for emergency or off-grid communication. Finally, it is interesting to investigate how Rhizome bundles can be prioritized or filtered, e.g., to distribute critical messages faster.

Acknowledgments

This work has been funded by the LOEWE initiative (Hessen, Germany) within the NICER project, and by USAID (USA), Radio Free Asia (USA), the NLnet Foundation (Netherlands) and AusAID (Canberra, Australia).

References

[1] D. M. West and M. Orr, "Race, gender, and communications in natural disasters," *Policy Studies Journal*, vol. 35, no. 4, pp. 569–586, 2007.

[2] L. Comfort and T. Haase, "Communication, coherence, and collective action: The impact of Katrina on communications infrastructure," *Public Works Management & Policy*, vol. 10, no. 4, pp. 328–343, 2006.

[3] P. Gardner-Stephen, "The Serval Project: Practical wireless ad-hoc mobile telecommunications," *Flinders University, Adelaide, South Australia, Tech. Rep*, 2011.

[4] P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd, "The Serval Mesh: A platform for resilient communications in disaster & crisis," in *IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE, 2013, pp. 162–166.

[5] P. Gardner-Stephen, A. Bettison, R. Challans, and J. Lakeman, "The rational behind the serval network layer for resilient communications," *Journal of Computer Science*, vol. 9, no. 12, p. 1680, 2013.

[6] P. Gardner-Stephen, J. Lakeman, R. Challans, C. Wallis, A. Stulman, and Y. Haddad, "MeshMS: Ad hoc data transfer within a mesh network," *International Journal of Communications, Network and System Sciences*, vol. 8, no. 5, pp. 496–504, 2012.

[7] P. D. Pradeep, B. A. Kumar *et al.*, "A survey of emergency communication network architectures," *International Journal of u-and e-Service, Science and Technology*, vol. 8, no. 4, pp. 61–68, 2015.

[8] M. Berioli, N. Courville, and M. Werner, "Emergency communications over satellite: the WISECOM approach," in *16Th IST Mobile and Wireless Communications Summit*. IEEE, 2007, pp. 1–5.

[9] A. S. Cacciapuoti, F. Calabrese, M. Caleffi, G. Di Lorenzo, and L. Paura, "Human-mobility enabled wireless networks for emergency communications during special events," *Pervasive and Mobile Computing*, vol. 9, no. 4, pp. 472–483, 2013.

[10] W. Wang, W. Gao, X. Bai, T. Peng, G. Chuai, and W. Wang, "A framework of wireless emergency communications based on relaying and cognitive radio," in *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2007, pp. 1–5.

[11] M. Manic, D. Wijayasekara, K. Amarasinghe, J. Hewlett, K. Handy, C. Becker, B. Patterson, and R. Peterson, "Next generation emergency communication systems via software defined networks," in *Third GENI Research and Educational Experiment Workshop*. IEEE, 2014, pp. 1–8.

[12] V. Mayer-Schönberger, "Emergency communications: The quest for interoperability in the United States and Europe," *Paper 2002-7, John F. Kennedy School of Government, Harvard University*, 2002.

[13] T. Pecorella, L. S. Ronga, F. Chiti, S. Jayousi, and L. Franck, "Emergency satellite communications: research and standardization activities," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 170–177, 2015.

[14] H. Chenji and R. Stoleru, "Delay-tolerant networks (dtns) for emergency communications," *Advances in Delay-tolerant Networks (DTNs): Architecture and Enhanced Performance*, p. 105, 2014.

[15] J. Thomas, J. Robble, and N. Modly, "Off Grid communications with Android meshing the mobile world," in *2012 IEEE Conference on Technologies for Homeland Security*, 2012, pp. 401–405.

[16] Y. Liu, D. R. Bild, D. Adrian, G. Singh, R. P. Dick, D. S. Wallach, and Z. M. Mao, "Performance and energy consumption analysis of a delay-tolerant network for censorship-resistant communication," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2015, pp. 257–266.

[17] H. Ntareme, M. Zennaro, and B. Pehrson, "Delay tolerant network on smartphones: applications for communication challenged areas," in *Proc. of the 3rd Extreme Conf. on Communication*. ACM, 2011, pp. 14–21.

[18] K. Heimerl, K. Ali, J. Blumenstock, B. Gawalt, and E. Brewer, "Expanding rural cellular networks with virtual coverage," in *10th USENIX Symp. on Netw. Systems Design & Implementation*, 2013, pp. 283–296.

[19] D. Johnson, N. Ntlatlapa, and C. Aichele, "A simple pragmatic approach to mesh routing using Batman," in *2nd IFIP Int. Symp. on Wireless Comm. and Information Technology in Developing Countries*, 2008.

[20] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *IEEE Int. Conf. on Technology for the 21st Century*, 2001, pp. 62–68.

[21] A. Battestini, V. Setlur, and T. Sohn, "A large scale study of text-messaging use," in *12th Int. Conf. on Human Computer Interaction with Mobile Devices and Services*. ACM, 2010, pp. 229–238.