Demo Abstract: 64Key - A Mesh-based Collaborative Plaform

Federico Amedeo Izzo*, Lorenzo Aspesi*, Alberto Bellini*, Chiara Pacchiarotti*, Federico Caimi*, Gianluigi Persano*, Niccolò Izzo*, Pietro Tordini*, and Luca Mottola*†

*Politecnico di Milano, Italy, † RI.SE Research Institute of Sweden AB
federico.izzo@mail.polimi.it

ABSTRACT

We present 64Key, a hardware/software platform that enables impromptu sensing, data sharing, collaborative working, and social networking among physically co-located users independently of their own hardware platform, operating system, network stack, and of the availability of Internet access. 64Key caters to those scenarios such as computer labs, large conferences, and emergency situations where the network infrastructure is limited in operation or simply not available, and peer-to-peer interactions are prevented or not possible. By plugging a 64Key device in one's mobile device USB port, an independent network is created on the fly, which users access from their own device though a web-based interface. In addition to default apps such as chat, file sharing, and collaborative text editing, 64Key's functionality may be extended through the run-time installation of third-party apps, available at a public app store. We demonstrate our proof-of-concept implementation of 64Key with multiple apps in a set of key scenarios.

ACM Reference Format:
Federico Amedeo Izzo*, Lorenzo Aspesi*, Alberto Bellini*, Chiara Pacchiarotti*, Federico Caimi*, Gianluigi Persano*, Niccolò Izzo*, Pietro Tordini*, and Luca Mottola*†. 2018. Demo Abstract: 64Key - A Mesh-based Collaborative Plaform. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SENSYS2018).* ACM, New York, NY, USA, 2 pages.

1 INTRODUCTION

The proliferation of diverse mobile computing platforms, along with increasing security and privacy concerns, complicate on-the-fly data sharing and collaborative working among physically co-located users. Heterogenous hardware, operating systems, and network stacks are difficult to interoperate. Stricter network management compounds the problem, in that direct peer-to-peer interactions are often prevented. In the absence of dedicated support, merely exchanging a file between co-located mobile devices running different operating systems may reveal as an ordeal.

Say, for example, a large crowd gathers at a scientific conference. Some users need to exchange a large set of data, but the network infrastructure is severely limited and also prevents peer-to-peer interactions for security reasons. Users then resort to the laborious practice of email attachments if possible, or to cloud-hosted file sharing services that work inefficiently because of the little available bandwidth. Similar issues arise in a number of different situations; for example, in computer labs or in emergency situations. Moreover, impromptu crowdsensing is often impossible to achieve in the absence of Internet access [1].

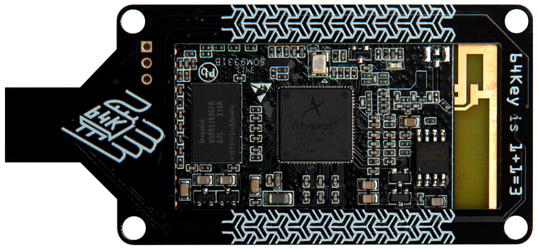

Figure 1: 64Key prototype.

Big industry players released technologies aiming at solving these issues, such as AirDrop by Apple and Near Share by Microsoft. These tools may be useful in specific scenarios, but they are not interoperable. Other solutions such as Resilio (formerly BitTorrent Sync) [3] only partly address the problem, as they require an existing network connection.

64Key is a hardware/software platform that precisely abates these limitations. Using 64Key, users plug a device like the one shown in Figure 1 into the USB port of one's mobile device. This creates a separate multi-hop mesh network spanning nearby 64Key devices, as intuitively shown in Figure 2. This network is independent of the hardware/software of the hosts, of the existing network infrastructure, and of the availability of Internet access.

Users access the apps running on the 64Key network in a platform-independent manner through a web interface from the host device. Our current prototype offers apps such as chat, file sharing, and collaborative text editing, as shown in Figure 3. Third-party apps may be downloaded and installed at run-time from an app store we make available. We explain next the 64Key hardware design, software architecture, along with our plan for future developments.

64Key is the result of a design process that embraces many of the concepts, practices, and tools found in the "maker" movement [5]. The team behind 64Key includes industrial designers, computer scientists, and electronics engineers. 64Key is thus significant also as an example of such design process.

2 SYSTEM DESIGN

Architecture. Each 64Key contains a small computing unit in the form of a SoM (System on a Module); this module includes a MIPS CPU, a WiFi radio and a device-mode capable USB interface. The computing module runs OpenWRT and contains the necessary software to operate the mesh network and to host the web-based 64Key

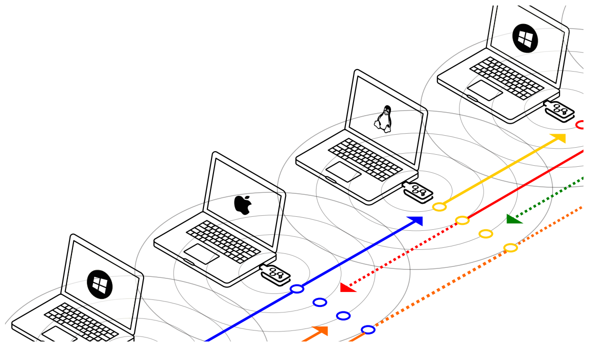

Figure 2: 64Key network diagram.

apps. The current prototypes consist in a custom PCB employing the OpenEmbed SOM9331 SoM [7], with Atheros AR9331 SoC, 64Mb RAM, 8Mb Flash, WiFi b/g/n, and a USB 2.0 port.

A network connection between the user's computer and the device is necessary to access the web-based interface. We use the RNDIS [4] protocol over the USB port, which is also used to power the device. Despite being a proprietary protocol, RNDIS ensures cross-platform compatibility.

The 64Key web-based interface is entirely hosted on the device itself and runs in the user's browser. This design choice has multiple advantages, such as not requiring any software installation on the host computer, ensuring compatibility with every modern browser, and letting users keep their own data on the 64Key itself. This feature makes the user data itself portable: data saved on the device is always accessible from different computers or operating systems. This feature is achieved without uploading anything to a public cloud, thus offering privacy benefits.

The web-based interface allows users to use the current 64Key prototype on any device with an available web browser, a host-mode USB port and RNDIS drivers, for example, any modern desktop or laptop and most Android devices.

Networking. The WiFi mesh network used by the 64Key is based on the batman-adv [6] layer-2 routing protocol. The batman-adv protocol provides connectivity and neighbor discovery. The latter functionality is provided as a RESTful API to 64Key apps that may need to discover nearby devices.

Providing unique IPv4 addresses in a decentralized network is non trivial. We employ a solution based on link-local [9] IPv6 addresses, which can be derived uniquely from the MAC address of each device. This results in every device obtaining a unique IPv6 address in a stateless way, without the need of a DHCP server on the network.

Routing inside the mesh network is handled by the batman-adv protocol; additional rules are set to forward packets from the 64Key web-application backend (running on the device) to the frontend (running on the user's browser).

The web-based apps hosted on the 64Key use the WebRTC protocol, which facilitates peer-to-peer connections by only needing a connection broker. The broker is decentralized and is based on the STUN protocol [8].

Downloading new apps from the app store uses the WebSocket connections between the client browser and the 64Key, to exchange Base64 encoded application files.

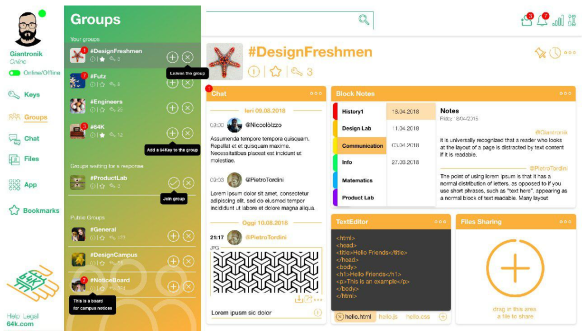

Figure 3: 64Key web-based interface.

3 DEMONSTRATION

Our demonstration will simulate an environment where no peer-to-peer or infrastructure-based internet connection is present, and two laptops with different operating systems need to communicate and cooperate.

Once a 64Key is connected to a computer, it boots in a few seconds, and begins to search for other 64Keys in range. When one or more devices are found, a IEEE 802.11s [2] WiFi mesh network is created.

The devices are recognized by the user's laptops as mass storage devices, containing just one .url shortcut file, pointing to the web-based 64Key interface hosted on the device itself. Opening the shortcut brings the user to the front page, where the available apps may be accessed.

We will demonstrate the following functionality:

I) group chat: an instant messaging chat, providing information about the other users, such as online status, last online time, personal message, and description.

II) collaborative text editing: cooperative text editor through which every user can create, edit, share text documents with other peers.

III) app store: where community-developed apps may be downloaded and installed on the 64Key, thus extending its functionality.

If at anytime during the activity of the 64Keys one peer of the mesh network disconnects, its status update is propagated over the network. The same happens for the connection of new users, which allows existing users to see all the available collaborators on the network in real time.

REFERENCES

[1] R. Ganti, F. Ye, and H. Lei. 2006. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine* 49, 11 (2006).
[2] G. R. Hiertz et al. 2010. IEEE 802.11s: The WLAN Mesh Standard. *IEEE Wireless Communications* 17, 1.
[3] Resilio Inc. 2018. Resilio: Faster file synchronization software with P2P & WAN optimization. https://goo.gl/LLcNHW
[4] Microsoft. 2014. Microsoft RNDIS Protocol Specification. goo.gl/KWeVzp
[5] I. Mohomed and P. Dutta. 2015. THE Age of DIY and Dawn of the Maker Movement. *GetMobile: Mobile Computing and Communications* 18, 4 (2015).
[6] Open-Mesh. 2018. B.A.T.M.A.N. Advanced Documentation Overview. goo.gl/NyCfX8
[7] OpenWRT Project. 2018. OpenEmbed SOM9331 v1. goo.gl/uWjuZm
[8] J. Rosenberg and otgers. 2003. *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs).* RFC 3489.
[9] S. Thomson et al. 2007. *IPv6 Stateless Address Autoconfiguration.* RFC 4862.