Scalable and Sustainable Community Networks for Inclusive Smart Cities

John Marlo Evangelista[1(✉)], Karyn Maglalang[1], Lope Beltran II[2], Colline Estrada[1], Pio Jonel Mijares[1], Ken Abryl Eleazar Salanio[2], Jhon Aaron Trajano[1,3], John Patrick Zamora[1], Vladimir Axl Von Carlo Zurbano[1], Isabel Montes-Austria[1], Cedric Angelo Festin[2], Matthew Podolsky[4], and Roel Ocampo[1]

[1] Electrical and Electronics Engineering Institute, University of the Philippines Diliman, Quezon City, Philippines
{john.marlo.evangelista,karyn.maglalang}@eee.upd.edu.ph
[2] Department of Computer Science, University of the Philippines Diliman, Quezon City, Philippines
[3] College of Computer Studies, New Era University, Quezon City, Philippines
[4] University of California Berkeley, Berkeley, USA

Abstract. Community networks can help build smart cities by interconnecting and empowering the underserved and unconnected. However, communities often face financial and human resource constraints when building and maintaining such networks. We present a network architecture and some design principles aimed at addressing issues of sustainability and scaling in the face of such resource constraints. With our proposed approach however, component selection, qualification, and testing become additional tasks for the community. Programmability and network function virtualization may instead help address compatibility issues, promote resource pooling at the edge, and allow dynamic resource allocation.

Keywords: Community networks · Software-defined networks · IPv6 transition

1 Introduction

Smart cities require an interconnected citizenry. However, in many economies, the digital divide remains a real and difficult challenge to solve, even in urban areas. One indicator that the problem still exists in a significant way is the disproportionate distribution of Internet access, which in turn may be attributed to several factors: lack of economic incentives for Internet service providers to invest in areas with potentially low returns, low purchasing power and related socio-economic factors in targeted areas [14], high infrastructure development and maintenance costs, difficult terrain and geography, and other factors not easily addressed by commercial or government-led initiatives [30].

S. Paiva et al. (Eds.): SmartCity360° 2021, LNICST 442, pp. 392–407, 2022.
https://doi.org/10.1007/978-3-031-06371-8_26

It is in this context that community networks continue to be an interesting proposition in addressing the problem of Internet access. There are certainly success stories worth replicating, such as the guifi.net effort [31] and many others [24] from which many lessons continue to be learned. A recurring and common theme is that unlike "top-down" and centrally-driven commercial or government-led efforts, community networks tend to be more grassroots-oriented, with varying models and degrees of organization and management [13]. However, it is precisely the grassroots orientation coupled with the socio-economic context that poses an additional challenge: community networks might indeed be a good solution in economically disadvantaged places, but these are exactly the places where the material resources and specialist skills needed to plan, establish, operate and sustain a community network may be in relatively short supply.

This brings us to two design goals worth considering at an early stage of conceptualization and planning: scalability and sustainability. In the context of this paper, we consider scalability as the ability of a community network to expand either (a) horizontally to accommodate more participants (users and providers), greater usage, or to enhance performance quality, or (b) vertically to be able to provide new and useful services. On the other hand, we define sustainability as the ability to assure continuous operation, management, and growth of a community network through the availability or application of necessary material and human resources as well as policy, social, and organizational frameworks. While there may be a myriad of other worthwhile design objectives and considerations, especially as may be gleaned from the rich experiences of relatively mature community networks [13], we focus on these two objectives that motivate our basic technical question: can an open and software-defined approach help build scalable and sustainable community networks, given the context and constraints in which these networks are situated?

This paper describes our modest 'work in progress' effort to address this question. We share the motivation, technical design principles, and initial experiences in deploying an experimental version of our Bayanihanets community network [25], now based on open and software-defined approaches. Our notion of 'open' refers to both hardware and software platforms with publicly available and extensible source code, designs, abstractions, or interfaces. On the other hand, 'software-defined' may somewhat overlap with the 'open' approach and will encompass the wide range of approaches including control plane - data plane logical separation [18], network programmability and standardized interfaces [22], network functions virtualization [15], network slicing [27], and similar technologies.

Specifically, we wish to explore whether an open and software-defined approach would:

- ensure better interoperability among devices, making it relatively easy for new participants, whether permanent or transient, to join;
- enable rapid deployment of either community- or externally-provided services, even 'deep' within the community footprint itself; and

– provide a more uniform view not only of the network but of user- and community-owned and managed devices as well, reducing the effort and expertise required to support and manage highly diverse and physically distributed hardware and software platforms.

This paper is further organized as follows: in Sect. 2, we pose some design challenges to community network deployment approaches vis-à-vis future needs for scalability and sustainability. Section 3 presents our proposed architecture, as well as results from evaluating candidate platforms and functional components. In Sect. 3.3, we briefly share some initial results as we tested these in two new experimental sites for Bayanihanets, our local community network effort. Finally, in Sect. 4, we conclude with further recommendations for current and future community network operators.

2 Designing Scalable and Sustainable Community Networks

2.1 IP Addressing in the Context of Scalability and Sustainability

It is neither scalable nor sustainable to provide Internet-routable IPv4 addresses to household customer premises equipment (CPEs) especially for community networks with limited Internet-routable IPv4 address pool. IPv6 provides not only larger address spaces but also the ability to use Internet-routable prefixes even within the community mesh itself, aside from link-local addresses. To ensure that interconnecting with the community mesh is a 'plug-and-play' experience without the need to configure addresses and routes explicitly, we use BMX6, an open-source distance-vector mesh routing protocol [26] on the mesh backbone and to join new access nodes, including home routers. BMX6 supports address autoconfiguration of participating interfaces by combining a default prefix (fd66:66:66::/64) with the EUI-64 suffix. BMX6 Unicast Host Network Announcements (UHNA) may be used to advertise IP addresses and internal home subnets at the option of the user, allowing community users to share internally-hosted services to the rest of the community. This paves the way for members to collectively host and support community services such as IoT services and public safety IP-based CCTV (closed-circuit television).

Not all end-devices, sites, or services on the Internet fully support native IPv6 yet, so we still need to support IPv4. Furthermore, some home users may prefer to keep whatever RFC 1918 IPv4 addressing scheme they already have. To allow users to join without needing to change their internal addressing schemes and to support IPv4-only devices and sites, we use 464XLAT [21] to allow traversal over the IPv6-only community mesh and provide network address translation services (provider-side address translation, or PLAT) at gateways to Internet uplinks.

The PLAT network address translation service needs a pool of Internet-routable IPv4 addresses. For community networks, having its own allocation of portable IPv4 addresses may be a challenge either because (1) it might be too expensive to secure membership and IP resources from the appropriate Internet registry, or (2) membership may require a legal entity to be in place, which might

not immediately apply to a nascent community network. For the time being, it might have to do with relatively fewer routable IPv4 address assignments from providers. This is an excellent motivating factor for setting up a community network: by purchasing transit Internet access in bulk, the community is better positioned to obtain concessions such as non-portable IPv4 assignments. We also believe that as IPv6 adoption increases on both end-devices, sites, and services on the Internet, the need for larger routable IPv4 pools for PLATs will hopefully decrease.

2.2 Open Platforms, Open Access

Open platforms. We believe the use of free and open-source platforms whenever possible is desirable for two reasons. Firstly, it promotes uniformity and interoperability in deployed and managed platforms, consequently allowing the development of a larger pool of collaborators willing and able to help manage the network. Secondly, open platforms promote community participation in design and configuration decisions: configuration settings, applications, and services may be published, reviewed, downloaded, or pushed in the most transparent manner. This is in stark contrast with the opaque, closed, and often secretive way in which many commercial providers manage customer premises equipment (CPEs) deployed for users.

As a specific example, a platform worth consideration is OpenWrt [10]. Community members may reflash commercially available compatible access points. OpenWrt APs and mesh nodes may natively support IPv4/IPv6 dual-stack, the 464XLAT IPv6 transition mechanism, and BMX6 routing for connectivity to the community mesh backbone. Community members may also share experiences selecting and reflashing commercially available access points, mainly because not all commercially available models are guaranteed to work with OpenWrt. Sharing local experiences and information regarding locally available products is invaluable in this regard.

Open Access. Although an orthogonal issue, we believe that open access may be a worthwhile, if not an altruistic design goal for community networks. If access to community services or the Internet is to be provided to transient or visiting users and capacity might be a concern, lower-than-best (LBE) effort QoS may be provided [25]. In turn, mechanisms like these may be deployed and supported more uniformly and transparently through the use of standardized yet free and open platforms. With Linux-based platforms, it is relatively straightforward to access and configure appropriate queue disciplines to provide LBE QoS to visiting users.

Aside from promoting shared learning experiences and knowledge pools, we believe that free and open-source platforms democratize access and participation in community networks by lowering the barrier to entry that might be posed by, say, standardizing to a single proprietary platform. Open platforms therefore may not only promote open access (basic usage) of the network and its services, but also open up access to operations, management, and decision-making, which in turn encourages better community participation in the management of the pooled resource [28].

2.3 Programmability and Virtualization from Access to Core

The use of programmable and virtualizable hardware, not only at the core but also towards the access edge, may also help with several aspects of scalability and sustainability. Network functions virtualization (NFV) may be used to address issues of compatibility, promote cost-sharing and resource pooling even at localized scales, and allow dynamic resource allocation. It may also help address the human resource challenges of running a network by reducing the need to master diverse and proprietary platforms, and by promoting service orchestration and network manageability at a greater scale. ISPs are potentially able to reduce annual operational expenses compared to legacy IPv4-based networks, thus providing sustainability and affordability for the community [16]. Moreover, the use of cheaper cost-effective commodity WiFi hardware makes SDN-based networks more flexible in their network architecture [29].

3 An Open and Software-Defined Bayanihanets

In this section, we discuss our application of the design principles and goals previously outlined to an evolved version of Bayanihanets, our approach to building community-based networks based on bayanihan, a local traditional and cultural concept of cooperation typically used within the context of community-based volunteer work [25]. The word "bayanihan" in turn is rooted in "bayani," which in the 1600s translated to "common labor or work" [23].

Our earlier work on Bayanihanets focused on the technical mechanisms needed to incentivize individual participation in community-based sharing of network resources by implementing a cooperative networking system using bandwidth aggregation, where owners enjoy the sum of their own bandwidth and whatever bandwidth they could borrow from other contributors. It was designed to make sharing more tangible by providing tools for managing, tracking, and measuring the network resources being shared, and the network resources being consumed while showing the impact of each user's activity on the overall network.

Our current work however evolves Bayanihanets further by incorporating the design principles and goals into its new architecture, presented in the next section. By software-defined Bayanihanets, we do not necessarily refer to the conventional use of frameworks such as OpenFlow, but instead we take the broader view of using programmable and dynamically-instantiated components and services within the network [19].

While other community networks focus on OpenFlow-based SDN approaches [12], our goal is to introduce an "open-access" architecture that will provide interoperability and programmability among the accessible devices in the market, making it relatively easier for community members to join the network. This will further help scale the community network to a wider set of cooperators. This is also made possible by transforming services into virtual network functions, allowing them to be installed in commodity hardware and used as virtualized customer premises equipment [17].

3.1 Architecture

Figure 1 illustrates the evolved architecture. We maintain the original features and functionalities of Bayanihanets, such as allowing users with personal Internet subscriptions to share their uplink capacity on a less-than-best effort (LBE) basis [25]. One of the new features is using a purely IPv6 backbone mesh. To support IPv4 on both the access side and the Internet, we used 464XLAT [21] through the Jool [6] network address translation package for stateless IPv4-to-IPv6 translation on the customer side translator (CLAT) and stateful IPv6-to-IPv4 translation on the provider side translator (PLAT) functions. Routing on the mesh backbone uses BMX6 for 'plug-and-play' functionality. To ensure support for both Jool and BMX6, we require CLAT user gateway devices and BMX6 meshing nodes to support Openwrt [10].

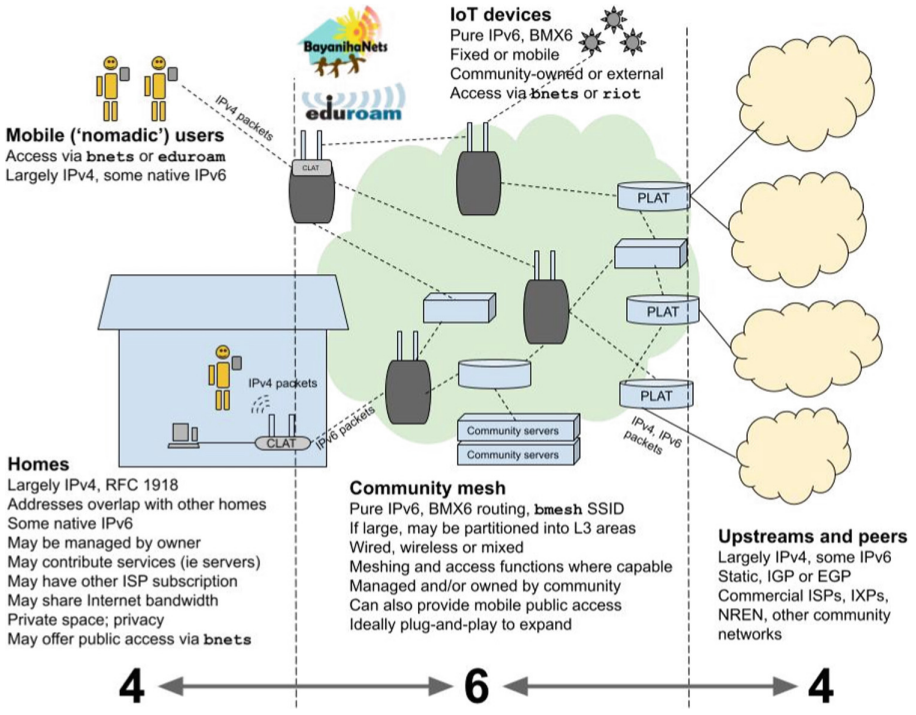


Fig. 1. Bayanihanets community network architecture

Prior to deployment, we went through a component selection process where candidate brands and models were tested. This is discussed in the next section.

3.2 Component Qualification and Selection

While it was relatively straightforward to draw up an architecture based on our design goals and objectives, mapping these to compatible, locally available, and crucially, relatively affordable components was another matter. There were inevitable conflicts between price, platform resources such as memory and processing ability (which would be crucial for programmability), support for open firmware and add-on software components, and performance. These are discussed in the following sections.

User Access and Backbone Mesh Devices. Since OpenWrt support was an important requirement, we searched for prospective user access and backbone mesh devices by first considering the cheapest and locally available Openwrt-compatible devices. We initially settled for a device marketed as an outdoor WiFi meshing platform with the following published specifications [1]. Some of the technical specifications of this platform, called 'Platform A' in the rest of this document, are listed in Table 1.

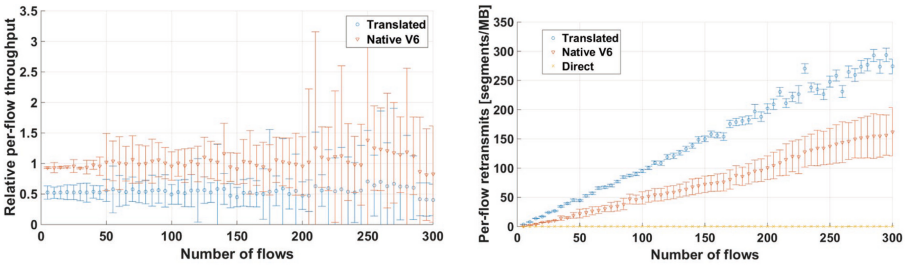
After installing Openwrt as its main firmware and installing Jool and BMX6 modules, Platform A was positioned as the device under test (DUT) in a simple *traffic source* $\rightarrow$ *DUT* $\rightarrow$ *traffic sink* logical configuration.

Table 1. Technical specifications of Platform A, targeted for user access and backbone mesh use.

Hardware resources	CPU 650 MHz, single core, 16 MB flash, 128 MB RAM Qualcomm Atheros QCA9531 SoC Qualcomm Atheros QCA9886
Wireless	300 Mbps 2.4 GHz 802.11b/g/n 867 Mbps 5 GHz 802.11a/n/ac 500 mW (27 dBm) max TX; −96 dBm RX sensitivity Dual external 5 dBi omnidirectional antennas
Wired interfaces	$2 \times 10/100$ Mbps Ethernet

User Access Device Performance. We wanted to see what users of "Platform A" might expect with respect to performance, particularly as a user access device performing CLAT functionality. We generated synthetic TCP traffic using the iPerf3 tool, mostly with default TCP settings (MSS = 1448), letting it seek a steady-state throughput based on the bottleneck bandwidth on the line.

From a macro perspective, the observed steady-state aggregate TCP throughputs across all flows were around 99.5 Mbps, 92.8 Mbps, and 52.2 Mbps for the no-CLAT-device, no-translation-required (i.e. IPv6-only flows), and translation-required (i.e. IPv4-only flows) cases, respectively. Figures 2a and b provide micro views on the impact on each flow as the number of concurrent flows increases. In Fig. 2a, we see that insertion of the CLAT device along the path results in



(a) (b)

Fig. 2. *(a)* Per-flow TCP throughputs, CLAT on "Platform A" relative to the 'no-CLAT' baseline case. This compares throughput performance for pure IPv6 (orange, no translation) and pure IPv4 (blue, with translation) flows through 100 Mbps network interfaces. All values are relative to the 'no-CLAT' average per-flow throughput performance. Error bars extending below the '0' y-axis value are no longer shown here. *(b)* Mean TCP segment retransmissions per unit volume of data transmitted, per-flow, as the number of concurrent flows increase. Points in yellow are for the control setup with no DUT in the path. Insertion of the CLAT device clearly results in increasing TCP retransmissions as concurrent flows increase, but to a larger extent when IPv4 translation needs to be performed. (Color figure online)

an insignificant drop in per-flow throughput when pure IPv6 flows, which do not require any address translation, traverse the "Platform A" CLAT function. However, when IPv4 flows are presented for translation, per-flow throughputs drop to around 50% of the no-translation-required (i.e. pure IPv6 flows) case. In Fig. 2b, we see that mean TCP retransmissions within flows increase linearly as the number of flows increase, indicating worsening packet loss or timeouts. Although this is expected as more flows contend for their share of the bottleneck bandwidth and as each flow exercises bandwidth-probing behavior, we observe worst-case behavior when the CLAT is burdened with address translation work on IPv4 flows.

What do all these mean in practical terms for users? Applications that tend to establish a large number of connections (such as BitTorrent) will degrade per-flow performance beyond the diminishing fair share of the bottleneck bandwidth. It will worsen with IPv4 flows. Users will therefore benefit if they favor IPv6 connections to servers and peers. It would also be advantageous if the community network provisions its local services and sites on IPv6 addresses.

'Platform A' as a Mesh Device. "Platform A" had also been initially positioned as a future wireless mesh backbone device, so a few preliminary tests were made on its performance, shown in Table 2.

These were by no means intensive tests and were just intended to quickly see if reflashing 'Platform A' to Openwrt would significantly impact performance as a meshing node. There was indeed some penalty, which suggests that further intensive testing and qualification would be needed for similar platforms

Table 2. "Platform A" comparative throughputs (in Mbps), stock firmware vs. OpenWrt. All tests were done with two nodes having line-of-sight over a 2.4 GHz wireless connection. No CLAT translation functionality was used in these tests.

Configuration	TCP		UDP	
	5 m	10 m	5 m	10 m
Stock firmware, bridged	92.5	93.1	96.1	96.0
OpenWrt, adhoc mode + BMX6	84.4	80.1	84.3	82.1

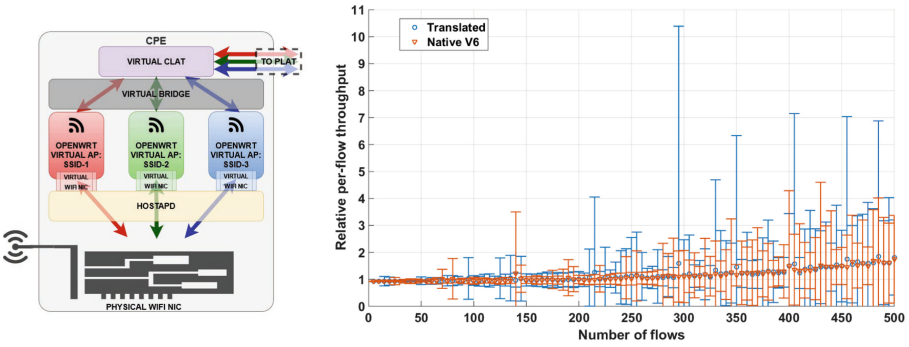
under consideration, perhaps using test environments that better replicate or approximate the intended deployment conditions.

Raspberry Pi as a User Access Device. We briefly tested the popular Raspberry Pi 4 device, equipped with a 1.5 GHz quad-core Cortex-A72 (ARM v8) and 4 GB of RAM. Such a device could function both as a CLAT+BMX6 gateway and as an application end-host as well. We inserted it into the same testbed previously used for 'Platform A' using a USB 3.0-to-Gigabit Ethernet adapter and its built-in Gigabit Ethernet port for connectivity and obtained 319.0 Mbps, and 276.1 Mbps average steady-state TCP throughputs for IPv4 (translated) and IPv6 (no-translation) flows, respectively. We have not evaluated it yet as a combined wireless access point and CLAT gateway. Nonetheless, it can quickly get users connected to use high-bandwidth services. For example, it could rapidly enable students to participate in videoconferencing and other remote learning activities.

Clearly, there are price-performance tradeoffs when selecting user access platforms. Table 3 provides a comparison of the two user access devices evaluated thus far. Community users may benefit from published comparisons not only to ensure interoperability with the local network but also to make informed decisions on platform selection, especially when cost is an important consideration.

Table 3. Price-performance comparisons, 'Platform A' and Raspberry Pi 4. Comparisons like these will help users make informed decisions on platform selection, especially when costs, no matter how relatively low, are an important consideration.

Platform	Local price [A]	TCP throughput		[C]/[A]	Remarks
		IPv6 [B]	IPv4 [C]		
Platform A	USD 37	92.8 Mbps	52.2 Mbps	1.41	Choose if providing wireless access is priority
Raspberry Pi	USD 88	319.0 Mbps	276.1 Mbps	3.14	Clear advantage in throughput and use for applications



(a) (b)

Fig. 3. *(a)* Virtual routers and access points as virtual network functions (VNFs) within host hardware shared by a number of users. *(b)* Wired throughput performance of a prototype VNF providing CLAT and BMX6 functionality. Note that these are for a 1 GbE network interface.

Localized Resource Pooling: Virtualized APs and CLATs. Community networks represent a form of network resource pooling that has been part of Internet tradition and architecture but on a reasonably localized scale. Our results from the previous tests of two potential low-cost client access and mesh platforms led us to the question: *would it be cost-effective to pool resources even at the access level?* That is, instead of a handful of households each having to search for an affordable yet compatible access device, would it make sense to deploy a more robust platform for shared use?

To test this approach, we prepared a KVM QEMU version 4.2.1 host on an Ubuntu 20.04.2 LTS (Focal Fossa) server with an Intel Core i7-3612QE 2.10 GHz 4-core/8-thread processor and 16 GB RAM installed. As shown in Fig. 3a, hostapd v2.9 [4] was used to manage the wireless network interface card and provide basic access point functionality. Virtual machines host OpenWrt-based virtual router and access point instances as virtual network functions (VNFs) servicing individual users. Setting aside the need for a prudent resource allocation scheme for the time being, we wanted to know whether the ability to statistically multiplex and occasionally overcommit resources in a flexible environment such as this may bring performance benefits. Figure 3b shows the relative TCP throughput performance of a CLAT router VNF provisioned with 8 vCPUs and 8 GB RAM. There seemed to be no significant performance penalty when the router VNF performs address translation for purely IPv4 flows compared to pure IPv6 flows that needed no translation. This shows that virtualizable, programmable hardware at the access edge has tremendous potential for resource sharing and dynamic resource allocation and provisioning.

VNFs need not exclusively offer OpenWrt-based virtual access point and router functionality, but may be used to flexibly and dynamically provision other functions and services at the edge as well. Platforms like these may also be used

to support third-party service delivery and infrastructure reuse, similar to the mobile virtual network operator (MVNO) model in the mobile telecommunications industry. Allowing the entry of multiple competing commercial operators offers potential revenue that may be used to sustain operations and expand the infrastructure. It also eases some of the burdens on the community itself to source and provide needed services.

Jumbo Frame Support. As a final note, devices to be used either as CLAT/BMX6 user gateways connecting to the mesh network or as mesh nodes themselves, must support jumbo frames with maximum transmission unit (MTU) sizes above 1500 bytes. We observed that many IPv4 sites on the Internet send IPv4 datagrams with 1500-byte MTUs and with Don't Fragment (DF) bit set. As these inbound 1500-byte datagrams were translated from IPv4 to IPv6, the larger IPv6 header size yielded a larger inbound packet which could not be accommodated on internal IPv6-only meshing links with the default 1500-byte MTU. Therefore, all IPv6-only links within the community mesh and all interfaces connected to these links should support MTUs above 1500 bytes. We found that in some cases, such as the Raspberry Pi 4, the Raspbian GNU/Linux 10 (Buster) operating system had to be recompiled to support jumbo frames. In other cases, the underlying hardware did not seem to support jumbo frames despite Openwrt compatibility.

Common Core Services. The core of the community network will host typical service provider functions such as shared routing gateways to transit or peering links, domain name servers, Web hosting services, authentication services, and others. We continue to advocate the use of virtualization and software-defined approaches to provision and manage these services.

An essential if not critical architectural component at the core of our community network design is the provider-side address translator (PLAT) that statefully translates N:1 IPv6 addresses to Internet-routable IPv4 addresses and vice versa [21]. As the network and its usage grow, the number and rate at which IPv6 addresses are presented for translation by the PLAT will necessarily grow as well. The consequential growth in the size of the translation tables coupled with the stateful nature of PLAT translation raises reasonable concern in the scalability of this function.

We wanted to determine the impact of an increasing number of flows on a candidate PLAT platform with a 1.70 GHz Intel Xeon 3104 CPU (6 cores) and 8 GB RAM. For this test, we used an iPerf generator machine equipped with 2×64-core CPUs running at 2.25 GHz nominal, 256 GB installed RAM, and two Intel x710 10 GbE network adapter cards. To minimize the possibility of the traffic generator itself becoming the bottleneck as the experiments progressed, we ran iPerf with the –zerocopy method of sending data [5], increased buffer sizes [8], and an increased number of RX/TX descriptors [7].

Figure 4 shows the median TCP throughputs for flows undergoing stateful translation by the PLAT, relative to 'no-PLAT' baseline conditions, for 500 concurrent flows and higher. It should be noted that these results are not directly comparable with results in Fig. 2a and Fig. 3b because the interface bitrates in

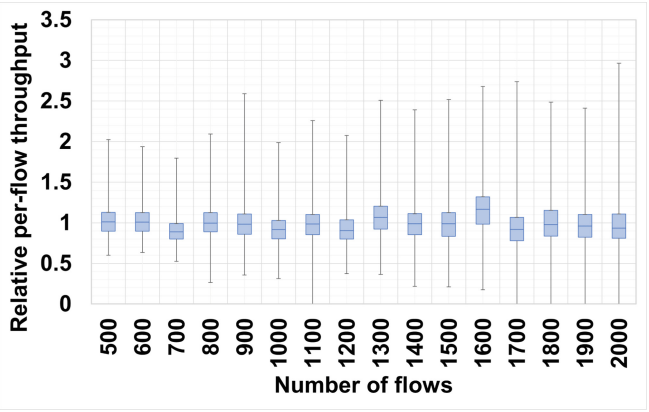


Fig. 4. Per-flow TCP throughput performance of candidate PLAT platform.

those cases were 100 Mbps and 1 Gbps, respectively. On the other hand, results in Fig. 4 are based on 10 Gbps interface bitrates. Steady-state TCP throughputs aggregated over all concurrent flows hovered around 3.43 Gbps with PLAT stateful translation. While it may be tempting to naively deduce that this PLAT platform might be able to handle translations from around 65 'Platform A' CLAT devices (equivalently: 65 households), with each presenting roughly 52.2 Mbps maximum worth of TCP IPv6 traffic for translation to IPv4 and vice-versa, clearly this should be validated by equivalently scaling the number of concurrent flows during testing. This might require testing with up to 20,000 concurrent flows and even beyond, which we have not yet been able to perform so far.

As a final note on testing: aside from scaling the number of concurrent flows, it is also important to note that packet processing rates rather than bitrate-based throughputs should be the metric to use. We have chosen to present results here in terms of bitrates simply to present the results in a more intuitive and relatable fashion to a broader and possibly even less technical audience. Finally, tests such as these are best performed using a mix of packet sizes that might better model actual Internet traffic [20].

3.3 Experimental Deployment

We applied the design principles and architecture described in the preceding discussion to pilot deployments in two residential housing clusters near our university's campus. We provided several 'Platform A' devices to some residents so that these could be further tested against real-world usage.

Figure 5 shows a few scenes from one of the two deployment sites. Aside from a local community-specific WiFi SSID, the network also supports the eduroam international WiFi roaming service to support students, teachers and researchers currently studying or working from home [3]. Figure 5 also shows results from the popular Ookla Speedtest online broadband testing service shared by one of the

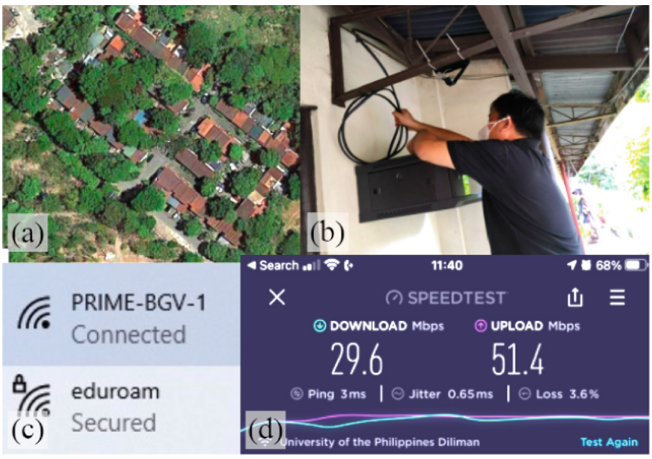


Fig. 5. Scenes from a pilot deployment. *(a)* Satellite photo of one of two deployment sites. *(b)* Installation and termination of fiber-optic uplink for community network. *(c)* Screenshot showing the local community WiFI SSID as well as the eduroam SSID. *(d)* Speedtest result shared by one of the residents.

residents – note how closely the 51.4 Mbps upload figure tracks the 52.2 Mbps laboratory test result for 'Platform' A in Table 3.

4 Conclusion and Future Work

Community networks can help accelerate the vision of building smart cities by interconnecting and empowering the underserved and unconnected. However, communities often face financial and human resource constraints that obstruct the scalability and sustainability of these networks. To address this problem, we presented a network architecture for community networks that stands on three design principles: (1) scalable, sustainable, and routable addressing, (2) open platforms and access, and (3) programmability and virtualization from access to core. There are well-defined Internet standards and mechanisms that can help with some of the technical aspects of community network sustainability, such as IPv6 and associated transition mechanisms. Additionally, there are free and open-source frameworks and other software components that allow the implementation of these technologies, but the commercial availability of compatible yet affordable hardware may be limited in some areas. This makes platform search, qualification, and testing a new challenging task for community networks. Even so, such an activity will be useful towards achieving technical scaling and sustainability and promoting cost transparency and informed platform choice among users. While not addressed in our current work, it will also be essential to include a security assessment of candidate devices and platforms.

Virtualization and programmable software-defined technologies at the edge may also address certain economic aspects of sustainability by enabling the entry

of third-party commercial entities not only as bulk Internet transit providers, but also as virtual network operators deep within the community itself. Towards this end, we hope to see better affordability and availability of platforms that support vCPE (virtual customer premises equipment) and uCPE (universal customer premises equipment) architectures [17], as well as the OpenWiFi initiative [9].

Finally, as resource-constrained communities strive to squeeze out more 'bang for the buck', that is, aim to achieve greater value for money from affordable commodity components, developing skill and expertise in platform tuning, optimization, low-cost acceleration, and similar techniques that used to be associated with high-capacity networks will become a necessity. Topics such as network interface card (NIC) tuning [8] and kernel bypass [2,11] may well become a staple of future training programs for future community-based network engineers, alongside more fundamental ones such as TCP/IP networking, wired and wireless network design and operation, and management of physical media and infrastructure.

Acknowledgement. The work described here is an extension of earlier work done in the Bayanihanets program funded by the Department of Science and Technology. The ongoing efforts are supported by the PCARI Scalable Community Access Networks (PCARI SCAN) and PCARI PRIME programs funded by the Commission on Higher Education, and the Asi@Connect CONNECT program funded by the European Commission through TEIN*CC. Pilot site deployment is done in collaboration with the University Computer Center of the University of the Philippines. Experimental resource support has been provided by Samsung R&D Institute Philippines. We are grateful to Eric Brewer of the University of California Berkeley for the valuable inputs and suggestions at the early stages of this work. Finally, we thank the APNIC Foundation for its support in disseminating our results.

References

1. Comfast EW72 1200mbps outdoor WiFi router repeater (2021). https://comfastwifi.us/comfast-cf-ew72-1200m-360degree-dual-band-5g-high-power-outdoor-ap
2. Data plane development kit (DPDK) (2021). https://www.dpdk.org/
3. eduroam for students, researchers, and educators (2021). https://eduroam.org/about/connect-yourself/
4. hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS authenticator (2021). https://w1.fi/hostapd/
5. iperf2/iperf3 (2021). https://fasterdata.es.net/performance-testing/network-troubleshooting-tools/iperf/
6. Jool: SIIT and NAT64 (2021). https://www.jool.mx/en/about.html
7. Linux tuning (2021). https://fasterdata.es.net/host-tuning/linux/
8. NIC tuning - Intel 1GE and 10GE NICs (2021). https://fasterdata.es.net/host-tuning/linux/nic-tuning/
9. OpenWiFi: an industry movement for accelerating Wi-Fi infrastructure innovation (2021). https://telecominfraproject.com/openwifi/
10. OpenWrt/LEDE project (2021). https://openwrt.org/about
11. The vector packet processor (VPP) (2021). https://fd.io/vppproject/vpptech/

12. Abujoda, A., Dietrich, D., Papadimitriou, P., Sathiaseelan, A.: Software-defined wireless mesh networks for internet access sharing. Comput. Netw. **93**, 359–372 (2015)
13. Antoniadis, P., et al.: Best practices guide for community networks. Technical report, Network Infrastructure as Commons (2019). https://www.netcommons.eu/index.html%3Fq=content%252Fbest-practices-guide-cns.html
14. Barela, M.C., et al.: Towards building a community cellular network in the Philippines. In: Proceedings of the 8th International Conference on Information and Communication Technologies and Development. ACM (June 2016). https://doi.org/10.1145/2909609.2909639
15. Chiosi, M., et al.: Network Functions Virtualisation: An Introduction, Benefits, Enablers. Challenges & Call for Action. Technical report, European Telecommunications Standards Institute (October 2012)
16. Dawadi, B.R., Rawat, D.B., Joshi, S.R., Keitsch, M.M.: Towards energy efficiency and green network infrastructure deployment in Nepal using software defined IPv6 network paradigm. Electron. J. Inf. Syst. Dev. Ctries. **86**(1), e12114 (2020)
17. Goemaere, P.: The evolution of network virtualization in the home. Technical report, Technicolor (2019). https://www.nctatechnicalpapers.com/Paper/2019/2019-the-evolution-of-network-virtualization-in-the-home/
18. Kim, H., Feamster, N.: Improving network management with software defined networking. IEEE Commun. Mag. **51**(2), 114–119 (2013). https://doi.org/10.1109/MCOM.2013.6461195
19. Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015). https://doi.org/10.1109/JPROC.2014.2371999
20. Lencse, G.: Benchmarking stateless NAT64 implementations with a standard tester. Telecommun. Syst. **75**(3), 245–257 (2020)
21. Mawatari, M., Kawashima, M., Byrne, C.: 464XLAT: Combination of Stateful and Stateless Translation. Technical report 6877, Internet Engineering Task Force (April 2013). https://doi.org/10.17487/RFC6877
22. McKeown, N., et al.: OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
23. Medina, I.R.: A reconstruction of Philippine social history from tagalog dictionaries and vocabularies, 1600–1914. Asian Stud. **21** (2013). https://www.asj.upd.edu.ph/index.php/archive/104-vol-21-april-august-december-1983
24. Micholia, P., et al.: Community networks and sustainability: a survey of perceptions, practices, and proposed solutions. IEEE Commun. Surv. Tut. **20**(4), 3581–3606 (2018). https://doi.org/10.1109/comst.2018.2817686
25. Montes, I., et al.: Tangible sharing, invisible mechanisms. In: Proceedings of the 2016 Workshop on Global Access to the Internet for All, GAIA 2016. ACM Press (2016). https://doi.org/10.1145/2940157.2940161
26. Neumann, A., Lopez, E., Navarro, L.: An evaluation of BMX6 for community wireless networks. In: 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE (October 2012). https://doi.org/10.1109/wimob.2012.6379145
27. Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J.J., Lorca, J., Folgueira, J.: Network slicing for 5G with SDN/NFV: concepts, architectures, and challenges. IEEE Commun. Mag. **55**(5), 80–87 (2017). https://doi.org/10.1109/MCOM.2017.1600935
28. Ostrom, E.: Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press (1990)

29. Qureshi, K.I., Wang, L., Sun, L., Zhu, C., Shu, L.: A review on design and implementation of software-defined WLANs. IEEE Syst. J. **14**(2), 2601–2614 (2020). https://doi.org/10.1109/JSYST.2019.2960400
30. Surana, S., et al.: Beyond pilots: keeping rural wireless networks alive. In: 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2008, San Francisco, CA. USENIX Association (April 2008). https://www.usenix.org/conference/nsdi-08/beyond-pilots-keeping-rural-wireless-networks-alive
31. Vega, D., Baig, R., Cerdà-Alabern, L., Medina, E., Meseguer, R., Navarro, L.: A technological overview of the guifi.net community network. Comput. Netw. **93**, 260–278 (2015). https://doi.org/10.1016/j.comnet.2015.09.023