Autoconfiguration for Faster WiFi Community Networks

Kelvert Ballantyne
Faculty of Technology and Trades
Algonquin College
Ottawa, Ontario, Canada
ballank@algonquincollege.com

Wahab Almuhtadi
Faculty of Technology and Trades
Algonquin College
Ottawa, Ontario, Canada
almuhtw@algonquincollege.com

Jordan Melzer
TELUS Communications
Ottawa, Ontario, Canada
Jordan.Melzer@telus.com

Abstract— **Commercial community wireless networks usually rely on wireless donors to give away Internet access while at home in exchange for access while they are mobile. The donors who make up the network get nothing immediate in return for providing Internet access. In this paper, we explore automatic configuration for a model of community networks in which donors mesh with each other and can aggregate Internet access from their peers with their own. To make the speed gains from access aggregation work for normal internet users, we use Multi-Path TCP (RFC 6824), proxying TCP traffic to MPTCP as needed. We examine the association, addressing, discovery, and routing challenges involved with self-configuring faster community networks, presenting IPv4 and IPv6 solutions to them.**

Keywords— MPTCP, Wireless mesh networks (WMN), Optimized Link State Routing (OLSR), 802.11s, Direct Subscriber Line (DSL), IPv6, IPv4

I. Introduction

First generation community WiFi networks were designed for coverage. These networks had few gateways to the Internet from which they provided access to a large community of users. All traffic from each user was usually routed consistently over one node out of the mesh network to the Internet. Commercial “community” WiFi networks re-use home Internet gateways routers as “hotspots” which provide WiFi access. These networks allow wired ISPs a “mobile” strategy or add capacity to cellular networks. Large Internet service providers such as British Telecom, Deutsche Telekom Iliad, SFR, Softbank, Comcast, and Cablevision have launched commercial community WiFi networks. In these networks, mobile users may access any of the operator’s home-based hotspots. The prospect of free access to WiFi hotspots while outside the house provides the incentive for the user to be a network hotspot donor. Previous related work in this area have tended to stay away from involving changes on ISP middle boxes in the network design due to the organizational effort it would take to get ISPs to Cooperate and work together. Our Community network is uniquely design to be implemented from an ISP level because we have the cooperation of an ISP, and our design is specifically for customers of a single ISP.

We introduced a multipath commercial WiFi network model which provides a benefit to users while they are at home as well as while they are out of it. In this model, hotspots are connected directly to the Internet like in other commercial hotspot networks, but may also provide Internet access to each other over WiFi like in first generation community networks [1]. This second access route can provide higher aggregate speeds and greater reliability. In providing faster, more reliable Internet in the home, this hotspot model provides a stronger incentive to prospective network hotspot donors. Increasing the value of community networks for the donor customers may help ISPs to grow the network’s footprint, increasing its ability to serve both mobile and at-home users. This paper explores solutions for auto configuration in these networks.

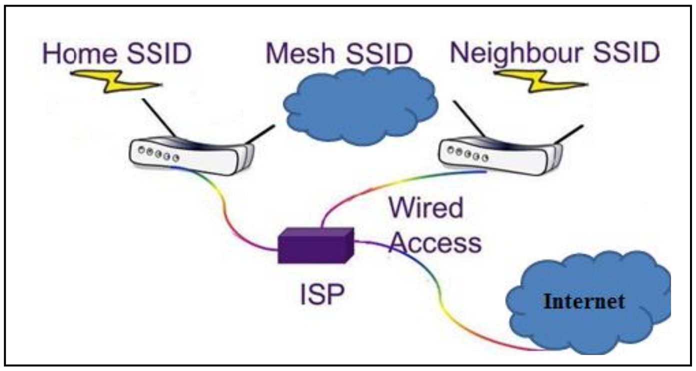

Fig.1. Community Backhaul Aggregation

Figure 1 illustrates wireless backhaul aggregation in a typical community or apartment complex. Neighbouring gateways, each having DSL Internet connections, use 802.11s to create a self-forming mesh network. At the transport layer, Multi-path TCP (MPTCP) aggregates the DSL and WiFi to neighour’s DSL Internet routes, making their use transparent to applications. The difference between conventional meshing scenarios and this model is that the mesh network does not just provide coverage extension through relay nodes, or redundancy for gateway nodes; it increases the peak throughput achievable by any user.

II. Purpose

Our goal is to create multi-pathing community WiFi networks that are self-configuring, requiring little to no involvement from users. The software will be responsible for building WiFi connections between neighbouring gateways. It will also handle the tasks of assigning and, where necessary, translating IP addresses, building routing tables suitable for aggregation, and transparently aggregating network traffic. This paper presents the challenges involved in autoconfiguration of link aggregation, mesh access, addressing and transparency as well as one coherent set of solutions to them which we have implemented in a testbed at the Wireless Research Lab of Algonquin College. The purpose of this paper is to give implementable guidelines that others may follower in order to replicate our testbed. We will not provide any performance evaluation data in this paper.

III. Link aggregation and opportunistic multipath

Multi-link protocols like MLPPP and LACP and multi-path protocols like ECMP are widely adopted, but they achieve link or network-layer aggregation only on interfaces with identical or known rates. As WiFi connection rates between neighbouring homes are not static, conventional aggregation techniques are not easily applied.

Multipath-TCP (MPTCP) [2] is an extension of TCP that adds the capability to send traffic over more than one path. While maintaining the TCP socket API to the application, a multipath TCP capable device can split a single TCP data-stream across multiple sub-flows, typically one per network interface. TCP congestion control (modified to maintain fairness with single-path TCP) determines the sending rate for each flow. By reusing TCP's existing ability to determine a sending rate, MPTCP can effectively use multiple paths with unknown capacities, providing both higher throughputs and higher reliability [3].

MPTCP uses new TCP options to signal multipath capability as well as to add interfaces and sub-flows. During the initial three-way handshake, MPTCP uses the "MP_CAPABLE" option to negotiate the use of MPTCP and establish keying material. If the negotiation fails, the client reverts to normal TCP behavior [4]. Any new sub-flow will also use TCP's three-way handshake and use the "MP_JOIN" option to identify the existing session and authenticate both sides. To be compatible with firewalls, most MPTCP implementations will see only the client initiate TCP subflows, creating one or more flows per client addressto each server address. The Université catholique de Louvain MPTCP implementation [14] in the Linux kernel creates one client flow per routing table with a route to the server address, per server address. In our testbed, in addition to dynamic neighbor discovery and address configuration which are done through existing tools, we use additional scripting to configure a separate routing table with a default route to the Internet for each path that a home gateway discovers.

Figure 2 below illustrates how MPTCP traffic differs from regular TCP traffic.

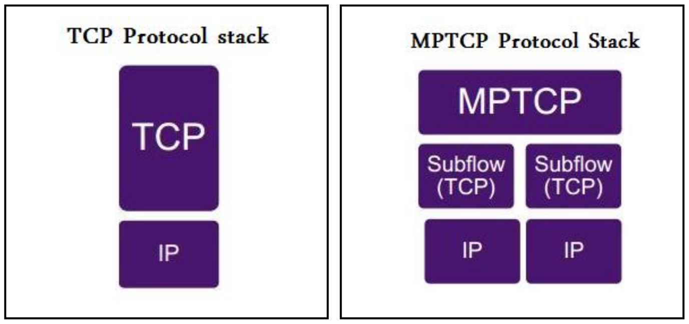

Fig. 2. TCP vs MPTCP protocol stacks

In our test bed, we used the Université catholique de Louvain implementation of MPTCP for the Linux kernel [5]. MPTCP across a direct Internet connection and a WiFi link demonstrated both aggregation and fast failover. MPTCP has previously been applied to aggregate or hand-over traffic between WiFi and 3G wireless interfaces [6]. Mobile device manufacture Apple has already implemented MPTCP on iPhones and iPads to support soft handover from home WiFi networks to mobile cellular provider networks for their Siri personal assistant application.

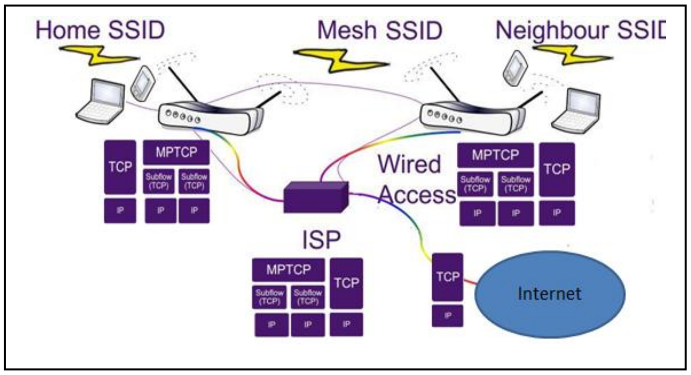

Fig.3. MPTCP Ethernet + Mesh Testbed

Figure 3 illustrates how TCP and MPTCP protocol stacks are incorporated into our test bed. As MPTCP is still experimental, the clients and servers are not MPTCP capable and run only TCP. Only the home gateway routers and ISP routers are multipath capable. TCP traffic from client devices connected to the WiFi access points is proxied to MPTCP at the home gateway routers, where it is split between a sub-flow over the wired link to the ISP and sub-flows that pass over WiFi to neighbouring gateways and through their wired links to the ISP. At the ISP routers, these MPTCP sub-flows are back to a regular TCP flow which connects over public Internet to the server. Traffic in the same TCP flow travelling in the reverse direction uses the same set of paths.

IV. WIRELESS MESH ACCESS

Wireless mesh networks (WMN) have been used for many years by community and enterprise wireless networks alike to extend the range of WiFi coverage. Wireless mesh networks were standardized in IEEE 802.11 through the 802.11s amendment, but many other meshing methods have been implemented over WiFi. It was important for the chosen wireless mesh routing protocol to be robust and able to self-heal without user intervention or long recovery times. The 802.11 mesh features enable wireless devices to self-form WMNs , is simple to configure, and offers link-level security options. Access points configured to the same channel and group SSID [7] can discover and neighboring nodes and build routes to more distant nodes. There is no central access point or path computation.

In our test bed – Fig. 1 – each home gateway router acts as a mesh station, configured with a common mesh SSID. Neighbouring gateways are served exclusively through the mesh SSID. To provide access to home devices, the device is also configured as a wireless access point. In an ISP network, this SSID would be managed by the home user. An ISP may also choose to operate additional SSIDs offering hotspot service to mobile users. We use an Open Source 802.11s implementation available in the mainline Linux kernel and maintained by the Open80211s.org project. While 802.11s is effective in building and autoconfiguring a mesh network, it does so at the link layer. It does not provide IP-level addressing or build IP routing tables. [8] To use MPTCP in combination with 802.11s to build a self-configuring network, other mechanisms are needed to configure IP addresses and routes.

V . Addressing and route building

When our multipath model of a community WMN is deployed the topology is not predetermined. Address auto configuration and neighbor discovery allow these networks to self-configure. In this short we present our IPv4 approach to addressing and routing using these tools. With limited address space each gateway router will obtain only one global IP address from an ISP. This means that the direct DSL interface will be the only one that has a global IP address. All IP addresses that sit behind this Internet gateway will have to be assigned from a private IP address space and NATed. This is the case for the WiFi access point clients, and also for the WiFi mesh interface. For clients of the WiFi access point, DHCP – the most common address assignment mechanism in IPv4 is sufficient for managing IP addresses and preventing duplicate addressing..

Addressing on the WMN, however, cannot easily be done through DHCP. All WMN nodes are peers, and there is not one central node chosen to manage and monitoring the network. Instead of having delegated globally reachable addresses or private addresses assigned by a DHCP server, the only addressing strategy available for the mesh interfaces is address auto-configuration using private IP address space. Each home gateway uses an auto-configured address inside the mesh and uses Network Address Translation to present all of the traffic it routes for other mesh nodes as coming from its own global address. For address configuration, we used avahi-autoipd to auto-configure mesh addresses. Avahi is a zero configuration protocol for IPv4 addressing. Avahi-autoipd runs independently on each device. It has service discovery and duplicate address detection mechanisms necessary for preventing the nodes from assigning an IP that is already in use by another node on the WMN. Figure 4 below shows how IP address are handled on a gateway's interfaces.

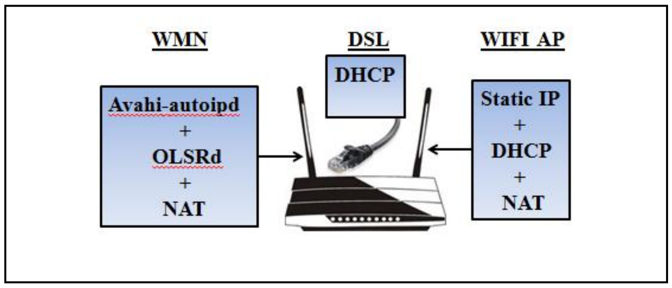

Fig. 4. Gateway Interfaces IPv4 Address Auto-configuration Methods

As neither 802.11s nor IP address autoconfiguration add IP routes as new nodes join the mesh network, we also need a routing distribution mechanism. In our testbed we use the olsr.org implementation of Optimized Link State Routing (OLSR) protocol. OLSR is an IP routing protocol for mobile ad hoc networks [9] that has proved effective on WMNs. As it does not directly support placing each new discovered neighbor into a separate routing table, in the olsrd.conf files, we assign new meshed nodes to a specific routing table that has no policy rules. From there, we use a separate task to build individual routing tables per neighbor discovered in order to expose these paths to MPTCP.

A. OLSRd Configuration

OLSRd, a mesh routing protocol daemon makes use of periodic hello messages on the WMN in order to discover single and double hop neighbor information. It performs a distributed selection of a set of multipoint relays (MPRs). Through frequent hello messages, a node may obtain up to 2 hop topology, detect neighbors, keep wireless sync and information for MPR selection processes [9]. When a node receives MPR information it recalculates and updates a standard routing table with each known neighbor, and because OLSR is a proactive routing protocol, routes to each one hop node within the WMN are made known immediately within the kernel routing table. This means that there is no route discovery delays associated when discovering new neighbors [9].

The open source OLSRd project effectively ran OLSRd on thousands of nodes with very little CPU power [10], proving that one may implement OLSRd on a large scale network. In our own system, each device in the mesh is a gateway router with direct Internet connection. Because of this, we are only concerned with 1-hop neighbors. As MPTCP includes a congestion control algorithm, we are able to minimize bandwidth and CPU power utilize by OLSRd by effectively creating ECMP routes across all meshed nodes, leaving link quality computations and path selection to MPTCP congestion algorithm. To achieve this we make use of the olsrd.conf file, in which we turn off OLSRd fisheye link quality algorithm and assign equal link quality and place equal cost value on all gateways. Bandwidth overhead and link congestion can be further reduce by increasing the interval at which OLSRd sends hello, topology control(TC) and MPR sync messages.

Routing and addressing may be accomplished in other ways. For example, MPTCP may be implemented with fewer routing tables by placing all mesh neighbours within a single table and using multiple flows and ECMP to allow MPTCP to discover and use all routes available to it. As the current implementation of MPTCP hard-codes a flow count, this approach may simplify routing at the expense of making the basic MPTCP setup more cumbersome.

VI. TRANSPARENCY

Full transparency means that user application will not be aware of MPTCP in action. This creates the need for a proxy mechanism in the home gateway software to adapt regular TCP traffic from client devices to MPTCP subflows on active gateway interfaces. In our testbed, we built an MPTCP to TCP proxy by running a stock transparent TCP proxy on a machine with an MPTCP enabled kernel. We used the Mallory Proxy server, a security focused “man in the middle” (MiTM) style fully transparent TCP proxy. Mallory intercepts all TCP traffic, acting as a server to clients and a client to servers.

As Internet server are also not generally MPTCP capable, it is necessary to terminate MPTCP traffic within the ISP network using a second on-path transparent proxy. Figure 5 illustrates our software topology. TCP traffic from client devices reaches gateway routers through WiFi access points, it is then redirected to Mallory acting as a MiTM which proxies TCP to MPTCP, creating subflows over direct DSL and closest neighboring mesh links. At the ISP router MPTCP traffic is terminated and packets are reordered and passed to the server over the Internet as a single TCP connection. Both client device and ser ver are unaware of MPTCP. Figure 5 demonstrates the various stages that TCP traffic passes through from client devices connecting to WiFi access points on gateway routers.

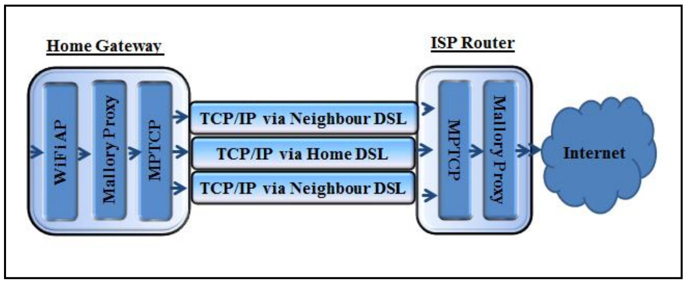

Fig. 5. TCP to MPTCP Network Traffic

VII. GATEWAY AUTOMATION SCRIPTING

Bash scripting is used to automate and start network programs and services. We also use scripting to check for new routing information from OLSRd and to build MPTCP compliant routing tables. For instance, we wouldn’t want Avahi-autoipd to start configuring an IP address on the mesh interface before the WMN is established. OLSRd will populate new default routes on the network to one Kernel routing table. Because MPTCP requires each default route to be on individual routing table as indicated a custom script is used to port default routes from the one table to various tables having policy rules base on source address. To achieved improved reliability, a connection manager should point the system “default” default route at a working connection.

VIII. CONCLUSION

Inspired by the rising popularity of home-based WiFi hotspot networks, we have developed a method for using a network of home WiFi gateways to increase the speed and reliability of Internet service to the home. Within our system, Internet-connected home gateways wirelessly mesh with each other using the IEEE802.11s WMN protocol. These gateways proxy TCP traffic from the home onto TCP’s emerging multipath variant, MPTCP, which is able to send traffic into the ISP network both over the direct connection to the ISP and through neighbouring home gateways. At the ISP, the multipath is resolved as the traffic is proxied back to TCP and sent over the open Internet to its destination. This commercial community WiFi design has proven implementable and effective on our testbed.

In future work, we plan to reduce the amount of custom scripting and the differences between the IPv4 and IPv6 implementations. We plan to reduce our depending on 802.11s by developing implementations based on 802.11’s ad-hoc or infrastructure modes, and also show that this model may be used with any combination of LTE, WiFi, and wired Internet. To validate the in-field potential of the system, we plan to do building-to-building performance measurements on modern, 802.11ac hardware.

REFERENCES

[1.] Wahab Almuhtadi, Jordan Melzer,Thanh-Hieu Nong, Ricky Wong, “Aggregating Internet Access in a Mesh-Backhauled Network through MPTCP Proxying”, ICNC 2014 “TCP Extentsions for Multipath Operation with Multiple Addresses” Retrieved September 26th 2014 from http://tools.ietf.org/html/rfc6824S. Venkat Mohan and N. Kasiviswanath, “Routing Protocols for Wireless Mesh Networks”, International Journal of Scientific & Engineering Research, Volume 2, Issue 8, Aug. 2011.

[2.] S. Barr , Olivier Bonaventure, Costin Raiciu, Mark Handley, “Experimenting with Multipath TCP”, SIGCOMM 2010 Demo, 2010.

[3.] “Multipath TCP – Linux Kernel Implementation: Release 86”. Retrieved July 10, 2014 from http://multipathtcp.org/pmwiki.php?n=Main.Release86

[4.] C. Paasch, G. Detal, F. Duchene, C. Raiciu and O. Bonaventure, “Exploring Mobile/WiFi Handover with Multipath TCP”, ACM SIGCOMM workshop on Cellular Networks (Cellnet'12), Helsinki, 2012.

[5.] S. Venkat Mohan and N. Kasiviswanath, “Routing Protocols for Wireless Mesh Networks”, International Journal of Scientific & Engineering Research, Volume 2, Issue 8, Aug. 2011.

[6.] “IEEE 802.11s” Retrieved September 30th 2014 from http://wireless.kernel.org/en/developers/Documentation/ieee80211/802.11s.

[7.] Y.-D. Lin et al.,“Indoor deployment of IEEE 802.11s mesh networks: Lessons and guidelines”, Ad Hoc Networks. (2011), doi: 10.1016/j.adhoc.2011.03.003.

[8.] Arun Kumar, Lokanatha C. Reddy, Prakash S. Hiremath “Performance Comparison of Wireless Mobile AdHoc Network Routing” Retrieved September 22nd from http://paper.ijcsns.org/07_book/200806/20080647.pdf.

[9.] M. Abolhasan, B. Hagelstein, J. C.-P. Wang (2009). “Real-world performance of current proactive multi-hop mesh protocols” Retrieved September 20th from http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1747&context=infopapers.

[10.] "OLSR-NG" Retrieved September 29th 2014 form http://wiki.funkfeuer.at/wiki/OLSR-NG.