Heilmeier's Catechism

What are you trying to do? Articulate your objectives using absolutely no jargon **We want to make neighboring routers work together for large file downloads, such that download time is significantly decreased.**

How is it done today, and what are the limits of current practice? **Currently, bandwidth aggregation is achieved by combining multiple interfaces into one logical one. These interfaces may be physical links combined at a switch, or TCP sub-flows which are managed at or near the receiver. In the latter case, packet reordering caused by out of order delivery is the biggest issue.**

What's new in your approach and why do you think it will be successful? **Bandwidth aggregation has always been the preferred option for those seeking to increase bandwidth without**

Who cares? If you're successful, what difference will it make?

What are the risks and the payoffs?

How much will it cost? How long will it take?

What are the midterm and final exam to check for success?

Needs - what are the needs of the project?

- **Faster download speeds for large downloads.**
- **Cheap DIY solution, with minimal infrastructure overhead**
- **A fair and safe protocol that handles issues of liability and privacy**
- **achieve bandwidth aggregation while side stepping concerns like packet reordering.**
- **A client cannot hog a neighbors bandwidth**

Approach - how do we achieve these needs?
Aggregate bandwidth by coordinating file download between nearby cooperating routers. The following will occur on a per request basis:

- **Routers will communicate via a defined protocol.**
- **The root router (who gets the original download request from a client on his subnet) will poll his peers to request their participation in a group download.**

- It will be up to each peer to decide whether they can afford to commit a percentage of their bandwidth to the download.

Proxy server to run on router and manage aggregation.

- For each incoming GET request, decide whether it is for a file download (uri parsing).
- Divide the file into [start, end] chunks that peers can download with HTTP range field.
- Manager thread that waits for response from peers and buffers (if it is ahead of the current file position), or sends to client

Security, fairness, and Liability

- A neighboring client should not have access to anything that the recipient client is downloading.
- Peer routers should only know WHERE to get the file, and WHAT part to download.
- A router will ALWAYS prioritize traffic from its own clients over peer clients.
- It must be clear from an ISPs perspective that potentially illegal traffic going to peer routers are actually for the client router.

Benefits - should the needs be met, what is the gain?

- Bandwidth aggregation realized using only existing infrastructure and open source software.
- Utilize unused bandwidth of nearby participating routers during off-peak hours.
- Application agnostic aggregation, compatible with any existing infrastructure or endpoint.

Competition - has this been done before?

- Bandwidth aggregation has been achieved.
- Various proprietary solutions already exist, each achieving the goals in their own ways?

BAMQP

Dan Robertson

September 2013

Abstract

It is often the case in networking that a desired throughput goal exceeded the limit of possible bandwidth. Although bandwidth speed increases 10 fold every few years, newer and faster speeds often come with a price hike that might not be worth the money. Fortunately, there exists a cheap solution that yields desirably higher bandwidth using only existing architecture and a few clever hacks. Link aggregation allows multiple network links to be combined into one logical interface with the combined bandwidth of the supporting links. This has been achieved at the lowest 3 layers of the OSI model, with both proprietary and open software solutions in existence that get the job done.

This MQP goes hand and hand with the ideas and practices behind link aggregation. We will see that while there are many solutions, there is little

1 Intro

2 Background, Related works

- Bandwidth aggregation is a novel practice frequently employed by network engineers who wish to supercharge their network's bandwidth without upgrading their links.
- In most cases, this means combining multiple outgoing physical links at a switch to create one logical link that offers the bandwidth equivalent to the sum of the outgoing links.
- Current work in this field targets aggregating different Radio Access Technologies (RATs).
- Transmission Control Protocol (TCP).

The practice of link aggregation was standardized in November 1997, by the IEEE 802.3 group. They recognized that Ethernet was most of the market, and that several vendors already had solutions for aggregating physical links to provide a faster Ethernet connection. However, an inter-operable solution was desired. The protocol

they developed became known as the "Link Aggregation Control Protocol"(LACP), officially as 802.3ad, but later renamed to 802.3AX. LACP allows networking devices which implement the protocol to send LACP packets to their similar peers in order to negotiate a link bundling scheme. Amongst the many goals of the LACP protocol was to provide dynamic link addition and deletion for the logical link, as well as fail over points when a link was unexpectedly severed.

One of the core principles behind link aggregation, and this MQP, is cost. Upgrading to a higher speed link is always an option, but trunking together multiple existing lower speed links can satisfy most, if not all, of the same needs. The added bonus is resilience, a failure of one link just means a decrease in throughput, instead of a total loss. It is important to note however, that Ethernet was the main focus during the time in which the LACP protocol came into being. It dealt with aggregating multiple physical links using a switch. Wireless link aggregation was never detailed in their report.

Link aggregation can be separated into two breadths. adaptive, which focuses on dynamically shifting traffic distribution methods based on mutating network characteristics, and Non-adaptive, which relies on purely static configurations.

2.1 Heterogeneous Wireless Networks - Bringing Link Aggregation to Radio Access Technologies

With the surging popularity of video streaming, high definition content, and online gaming, bandwidth limits are again becoming a problem. Aggregating several Radio Access Technologies (RATs), such as LTE and WiFi. Multinode terminal devices (such as smart phones) have the power to actively switch between RATs, but to use both simultaneously is a possibility not currently in use. This is because of a few major set backs. First, for real time and TCP applications, splitting traffic between links with different latencies can result in out of order delivery, which adversely affects TCP throughput. Second, battery power usage becomes an issue when multiple RATs are being used simultaneously. This MQP is less concerned with battery usage, but the issue of packet reordering is at the forefront of any of these techniques.

- Combining multiple RATs can yield throughput equivalent to the sum of the individual lines. Helps reduce load on a particular link by evenly distributing it over many links. Distribution order must preserve packet order.
- Packet reordering outside a certain sequence range will trigger a TCP loss event, which shrinks the transmission window and negatively affect throughput.
- Packet reordering metrics: Reorder Density (RD) and Reorder Buffer-occupancy Density (RBD).

- RD - distribution of out of order packets, normalized to the number of packets, for any given sequence.
- RBD - measures the buffer occupancy frequencies normalized tot he number of non duplicate packets. Good for predicting amount of resources required to preform reordering.

2.2 A solution for every layer

Bandwidth aggregation has been realized on all network layers except the Physical. These solutions each have their advantages and disadvantages, which are summarized below.

2.2.1 Application Layer

Here, the application has access to multiple outgoing interfaces, and can split the data into several application layer chunks (as this MQP attempts to do with HTTP) which it transmits simultaneously over these interfaces. The obvious issue here, is that the aggregation must be application specific, and does not provide more general, application agnostic aggregation. Examples include XFTP, MuniSockets and Parallel Sockets. They all provide similar solutions that combine multiple TCP connections into one logical one.

2.2.2 Transport Layer

Multipath TCP (MTCP) has been defined by the IETF in 2011. It breaks outgoing data from the application layer into multiple streams that can travel out different interfaces. Segments that share an interface are considered sub flows. It doesn't have support for some of the more desirable bandwidth aggregation techniques, such as intelligent interface selection. To deal with reordering, MPTCP maintains a buffer for out of order packets that are kept until the can be reordered into the stream.

TCP isn't inherently optimized for timely delivery, so using it as a mechanism for aggregation only provides so much gain. Further, new TCP protocols will not be interoperable with existing network infrastructure.

2.2.3 Network Layer Solutions

IP protocol is easier to leverage to obtain bandwidth aggregation. Unfortunately, it is prone to more out of sequence arrivals.

The Round Robin packet scheduling algorithm, which runs in O(1) time, is a simple preexisting network layer approach to bandwidth aggregation. Packets (assumed to be of the same size) from the same flow are assigned to multiple paths in a circular matter. The RR scheduler only works so well in theory because it assumes homogeneous packet size and transmission rate. This is very rarely the case however, so the approach falls short.

Kim et al. (2008) proposed a BA scheme that employs two metrics for scheduling, bandwidth estimation and packet partition scheduling. The former determines the amount of bytes that can safely be transmitted across a link without triggering congestion.

Issues with round robin: the bandwidth may be limited to that of the slowest path.

2.2.4 Data link Layer

The data link layer was what most of the older solutions to bandwidth aggregation used. It was not until the mobile network boom that attempts to aggregate heterogeneous RATs at the link layer began. One such realization of this practice is the Generic Link Layer (GLL) protocol. GLL is a multi access radio architecture that aims to unite differing RATs under a transparent session. GLL allows for Multi Radio Transmission Diversity. "MRTD is defined as the data flow split (on IP or MAC PDU level) between two communicating entities over more than one RAT. This comprises parallel transmission, or dynamic switching between the available RATs."[**GLL:2005**] MRTD can select its links based off criteria such as traffic requirements and estimated link quality. However, like many others, this approach to bandwidth aggregation struggles with the issues of packet reordering and unbalanced load distribution.

2.3 Adaptive bandwidth aggregation

The pitfalls of the naieve approach (non-adaptive bandwidth aggregation) manifest themselves at every layer. Static decision making for link selection ultimately lead to out of order delivery of differing data segments, which must then be addressed by packet reordering. Adaptive bandwidth aggregation considers the varying link and traffic conditions when organizing sub-flows between multiple interfaces.w

Some solutions used at various layers:

- Luo et all (2003) Split traffic into important and non-important stream, delivering each stream over a different RAT. A small training packet is sent along each link to calculate the RTT and delays, this is done intermittently and allows their implementation to better split traffic between different links.

- Splitting traffic into streams is often done with video content, where a base layer (necessary for decoding the video) is separated from different high level enhancement layers, which enrich viewing of the video.

- W-SCTP, an extension of the stream control protocol, which uses a bandwidth aware scheduler (implemented on the sending end) to manage subflows across multiple paths [Casetti and Gaiotto (2004)]. It maintains separate send buffers per interface, and sends each segment down the fastest one until each path has reached its congestion window limit. This approach does not deal with segment reordering, and fails to use the vast majority of metrics for optimal link selection.

- Also see LS-SCTP, which achieves one logical congestion window made up of the aggregate of the participating paths.
- Multipath TCP is a big contender. Arrival time matching load balancing (ATLB) [Hasegawa et al. (2005)], presents a solution to the ordering problem. ATLB scores each path by end-to-end delay, and uses this heuristic (lower is better) to assign each segment to a path. This significantly cuts down on out of order delivery, but a reorder buffer at the client is still maintained to deal with infrequent reordering issues.

3 Approach

3.1 Deciding on a Device

- We knew we needed to pick the right hardware for the job, as this project could be implemented on any variety of devices that can emulate a router.
- The device needed to perform well under normal networking conditions, but needed to be extensible such that device restrictions would not impede the successful implementation of bandwidth aggregation between neighboring nodes.

We narrowed down the scope of our device search to three major families of devices: routers, single-board computers, and Linux computers. The first of these is the Raspberry Pi. Small, cheap, and robust, with a personalized flavor of Linux and support for just about any framework desired, the Pi was an attractive choice. It can be modified to meet a wide variety of needs, one of which includes emulating a router. The pi can supports a growing number of programming languages and boasts 2 GB of free disk space on the cheapest model. However, with a limited number of outgoing ports (2 USB and 1 Ethernet), the quality of router this device could mimic is not so favorable.

Second, a conventional Linux box could have been used. There is a large variety of network purposed Linux distributions and debian derivatives of various capabilities, which could be installed on any Linux platform. This could enable us to use multiple network interfaces and a powerful CPU with plenty of memory, which could perform well as router. This flexibility makes the combination of a robust distro and powerful hardware a hard option to ignore. The catch however, is the gluttony of the machine. Although the goal of this project is to provide a proof of concept prototype for our model of bandwidth aggregation, a cost effective implementation never hurt anyone.

The third possibility was to use an existing wireless router and flash it with open firmware that would allow it to be more easily altered. The configuration options provided by a commercial router out of the box would not allow for low level modification, and bandwidth aggregation cannot simply be achieved through altering a few NAT tables, so low level modification was a necessity. Because of an oversight of the

GNU public license, Linksys developers were force to make their WRT-54G drivers completely public in order to comply with the license. This opened up the door for a variety of hacked WRT firmware to be developed, eventually leading to DD-WRT.

DD-WRT is a buzzword amongst do it yourself networking enthusiast. It provides a large amount of functionality not present on the original router firmware, most of which is exposed in its configuration options. While the added functionality was nice, what we were more concerned with was the ability to load and run custom applications onto the router. For this, OpenWRT (a cousin of DD-WRT) would come in handy.

OpenWRT functions similar to DD-WRT, but it has more developer support and a livelier community then that of the DD-WRT. It comes preloaded with a package manager that makes installing custom software easy, which allows for the creation of C/Python packages and even Kernel Modules. These add-ons can be flashed onto the router as executables, or bundled with the original firmware image.

3.2 Implementing on a Proxy

4 Evaluation

5 Discussion

6 Conclusion, Future work

Todo list