Motivations

Some of these are core motivations, others are new motivations arising from the initial design work

- Multipath TCP - how does the OS know how to load balance while respecting ISP contract limitations on the user? **[Thanks Peter Tonoli]**
- “Apps” cannot currently be designed to fully respect ISP contract limitations. This impacts the poor and the technically illiterate the most
- How can VoIP and other real-time services be prioritised fully across the internet?
- How can ISPs shed load smartly during high congestion times?
- How do internet users learn more about the KPIs of their chosen ISP?

Design

Design Problems and Individual Solution Options

- Discovery of ISP endpoint to query -
    - IP
        - A reserved IP address (or IP address block) which the ISP always routes internally.
    - [Not feasible] DNS
        - Endpoint by convention - a reserved DNS hostname which cannot be used on the general internet, that the ISP can override
        - URL convention using “ISP” tld. A TLD just for ISPs to resolve DNS with - eg. Api.customer.isp
        - Users can bypass ISP DNS though - Not ideal. The home router will receive the DNS value, but then that needs to be passed downstream to devices. Not feasible.
    - [Not feasible] DHCP field which carries the data - this will stop at the home router though, and that means having to chain this from the home router downstream. Not feasible.
- Communication Protocol - Use of HTTPS / RESTful API - mainly using GET methods and URI params (which are easier to test and understand directly with a browser and address bar)
- Security and Privacy - should be TLS (HTTPS)
- ISP Infrastructure costs
    - This would be an API on top of data that is already being collected, so the impact is low.
    - Ideally, a low-power and low-cost ARM-powered device could be available for purchase which includes software and automatic updates from a given vendor.
    - The ISP still needs to set up routing, and integration with their internal usage tracking software. (Ideally, the vendor of that software would sell this API software)
    - The ISP might be allowed to get data back from consumers. This requires much deeper though to contrast customer privacy with useful industry data.
- Customer Authentication
    - IP address affinity - this means that a connection from a particular customer IP or IP Block is correlated with a customer ID. Such mapping requires information from other ISP systems.
- An algorithm to determine if the user will exceed monthly usage
    - This would be left up to the OS. Default policies can be put in place which can optionally be adjusted by the user.
    - A software API would be provided to apps

Enablers, and new features/opportunities

- See reserveRealtimeBandwitdh / reserveUnlimitedVolume below. This would add a lot of value for ISPs.

- The user device would send device information by default, but that can be disabled. This would give ISPs and the industry much more information about usage within the home for network planning and service/product development.

OS API (Thinking of Android Java SDK)

- isISPInformationRequestAvailable
- registerDevice - the device sends device information and gets back an ID to use to communicate with the ISP server
- registerAnonymousDevice - the device doesn't send any information, and the ISP server responds with an ID
- getDailyAverageBudget
- getDailyRemaining - projected or pro-rated if billing cycles are different. The ISP can indicate a "reasonable" amount according to
- getMonthlyRemaining - projected or pro-rated if billing cycles are different
- isLegalUnlimitedVolume - the user truly has unlimited download volume as per contract.
- isMarketingUnlimitedVolume - the user technically doesn't have unlimited download volume
- getMaxSyncDownRate - the max sync rate the ISP sees
- getMaxSyncUpRate - the max sync rate the ISP sees
- getMaxDownloadRate - the minimum value when comparing the download plan rate, and the download sync rate
- getCurrentISPCongestion - the minimum value when comparing the upload plan rate, and the upload sync rate
- reserveDownloadVolume - tells the OS that the app plans to download a given amount of data.
    - The OS can approve or deny, depending on policy.
    - The app can specify whether it's user initiated or background (low-priority) initiated, etc...
    - The response from the OS can also result in cancellation of approval later (due to other non-reserved usage taking up too much volume)
- reserveUploadVolume - similar to download
- reserveRealtimeBandwitdh - allows the ISP router and home router to coordinate priority policies (which don't always work well when only controlled from the home router side only). The user would reserve bandwidth for VoIP phonecall, Skype/Video, Gaming and for Video on Demand. The requester would send information about upload and download requirements. Upon request, the ISP software can talk to the lower level protocols and routers to prioritise the traffic so it's less likely to drop packets vs normal traffic. More details TODO.
- reserveUnlimitedVolume - similar to Realtime Bandwidth in the last point, this allows an app to use a very small amount of bandwidth (low-priority class) in return for "unlimited" volume. This would be very useful for apps like Dropbox. Download volume using this reservation class would not be metered. The ISP would respond with socket header details to include, but also the amount of maximum bandwidth allocated. When congestion is high, this class of connection would have packets dropped. More details TODO.