Modeling DSL with NetEm

DANIEL MOSS

Abstract

- With the increased use of internet based applications requiring low latency, and high bandwidth, the performance demands of the last mile network continue to grow. Additionally, the highly variant deployment scenarios of these technologies, have a high impact on their performance, creating difficult to replicate environments for application developers to test in, often requiring expensive and difficult to obtain equipment. This thesis attempts to model the networking performance of DSL using the open source tool NetEm. This is done by studying the latency performance of DSL connections under a range of conditions and configurations, to quantify the performance. That performance data can then be used to create delay models for using NetEm's custom distribution delay models, providing a powerful tool to test devices and software under simulated DSL conditions.

Initial Idea

- Want to reproduce the network performance of DSL connections
  - Shouldn't involve any specialized equipment
  - Should use open source tools
  - Should provide better modeling than typical testing methods

Research funded in part by Google.

Why Model DSL?

- DSL is an expensive to operate technology in a lab environment
  - Requires CO side equipment (DSLAMs)
    - These can be hard acquire (not commercial products) and very expensive
- DSL is the most popular Broadband technology world wide
  - Up to 81% of US homes have DSL available as an option
  - Utilization of DSL is ubiquitous in places like the UK, and very popular in other parts of Europe
- DSL is very complex
  - There are a massive amount of tunable parameters in a DSL connection
    - Each of these could affect the network performance of technology running over DSL.

Why use NetEm

- Open source readily available tool to Linux installations
- Well studied by others
- Easy to use and configure
- No special equipment required (meaning any models would be usable by anyone who needed to)

Our Problem and Hypothesis

- We need to create an accurate enough model to use in place of DSL using NetEm.
  - Needed to look at NetEm features to see what it can do
  - Needed to measure DSL to see how it looks under various scenarios
- Our Hypothesis
  - Bandwidth is a tightly controlled and predictable parameter
  - Latency is the real key standout of DSL
- Suggested solution
  - Study the latency of DSL under multiple scenarios, and focus on modeling that.

Basics of DSL

- To understand how it's modeled, first we need to understand DSL
  - Runs over copper cables (twisted pair) over the "last mile" into a home
  - Two pieces of equipment involved
    - DSLAM (CO side) Service provider deployments
    - Model (CPE side) Customer homes
  - Range locked technology (longer loops = worse performance)
    - Generally operates over 0 - ~23000ft depending on variety
    - Rates up to 200 Mbits+ in best case (35b, short loops)

Basics of DSL 2 (Equipment)

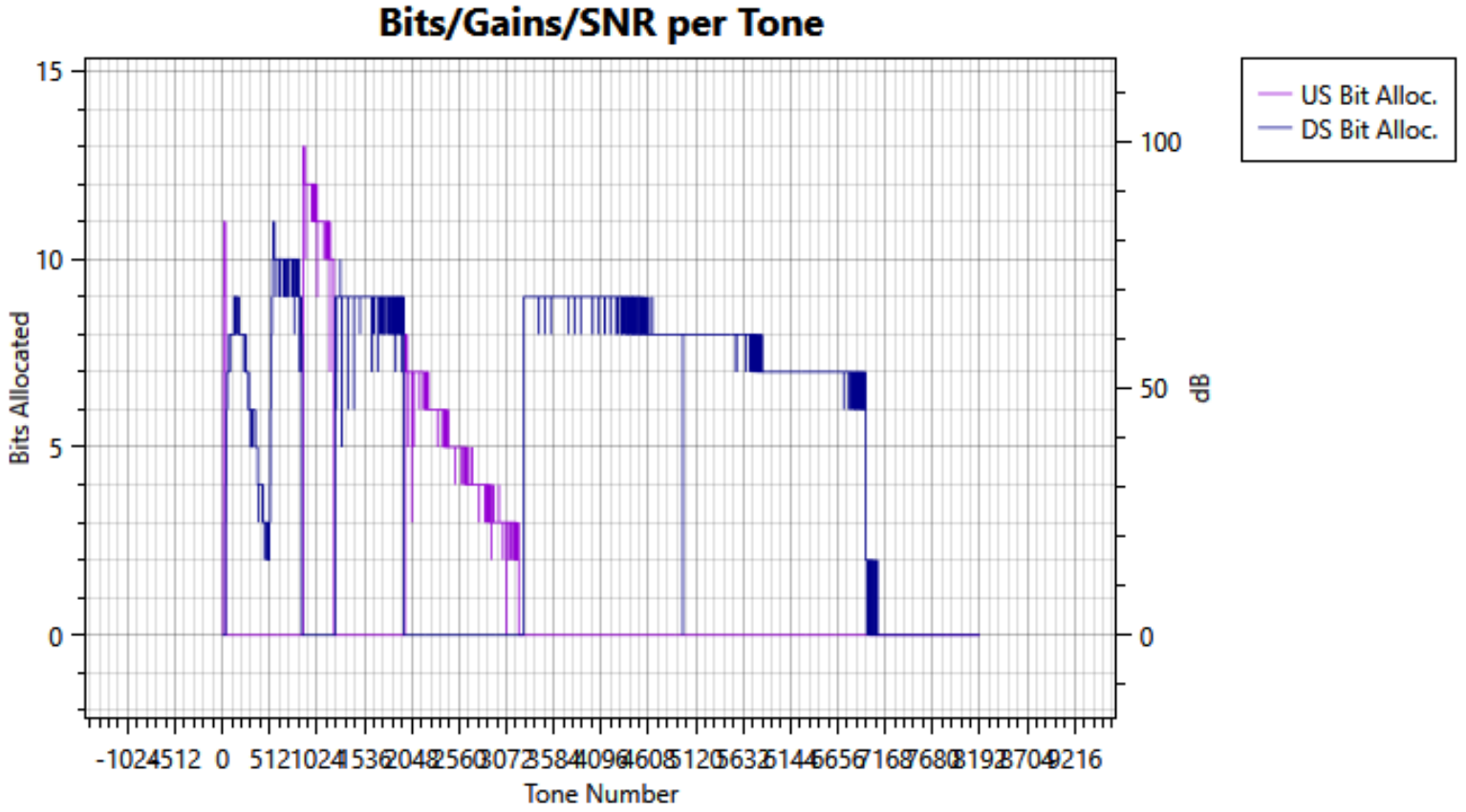
- DSL (Digital Subscriber Line), is a Digital signaling technology
  - Data is transmitted digitally between two chipsets
- DSLAM (Digital Subscriber Line Access Multiplexer)
  - Essentially a collection of 24-48 modems
  - Takes one or a few larger connections (generally fiber) and multiplexes to each customer
  - Allows for configuration of each customer line with a complete range of options
- CPE (Customer premise equipment) – a modem in your home
  - Generally a simple modem or gateway in a home
  - Usually provided by service provider
  - Single modem, less configuration typically

Basics of DSL 3 (Varieties of DSL)

- DSL has multiple varieties (Incomplete list, but major players)
  - ADSL (Asymmetric DSL) – Slow < 10 Mbit ds/1 Mbit us, Long range
  - ADSL2+ - Slow, but faster 3.5/24 Mbit US/DS, Long range
  - VDSL2 – We studied this!
    - Faster – up to 200+ Mbits depending on variety and loop
    - Multiple Bandwidths up to 35 Mhz
    - Many optional features (Retransmission, Vectoring)
  - Bonding can reach even higher rates
- Other forms of Symmetric DSL exist, but not as widely deployed

Basics of DSL 4 (Frequency)

- Transmission divided into Upstream and Downstream Bands
  - Amount and width of bands depends on configuration
  - Frequency domain duplexing technology (meaning both sides talk at the same time, just in different locations on the frequency band).



Basics of DSL 5 (Initial startup)

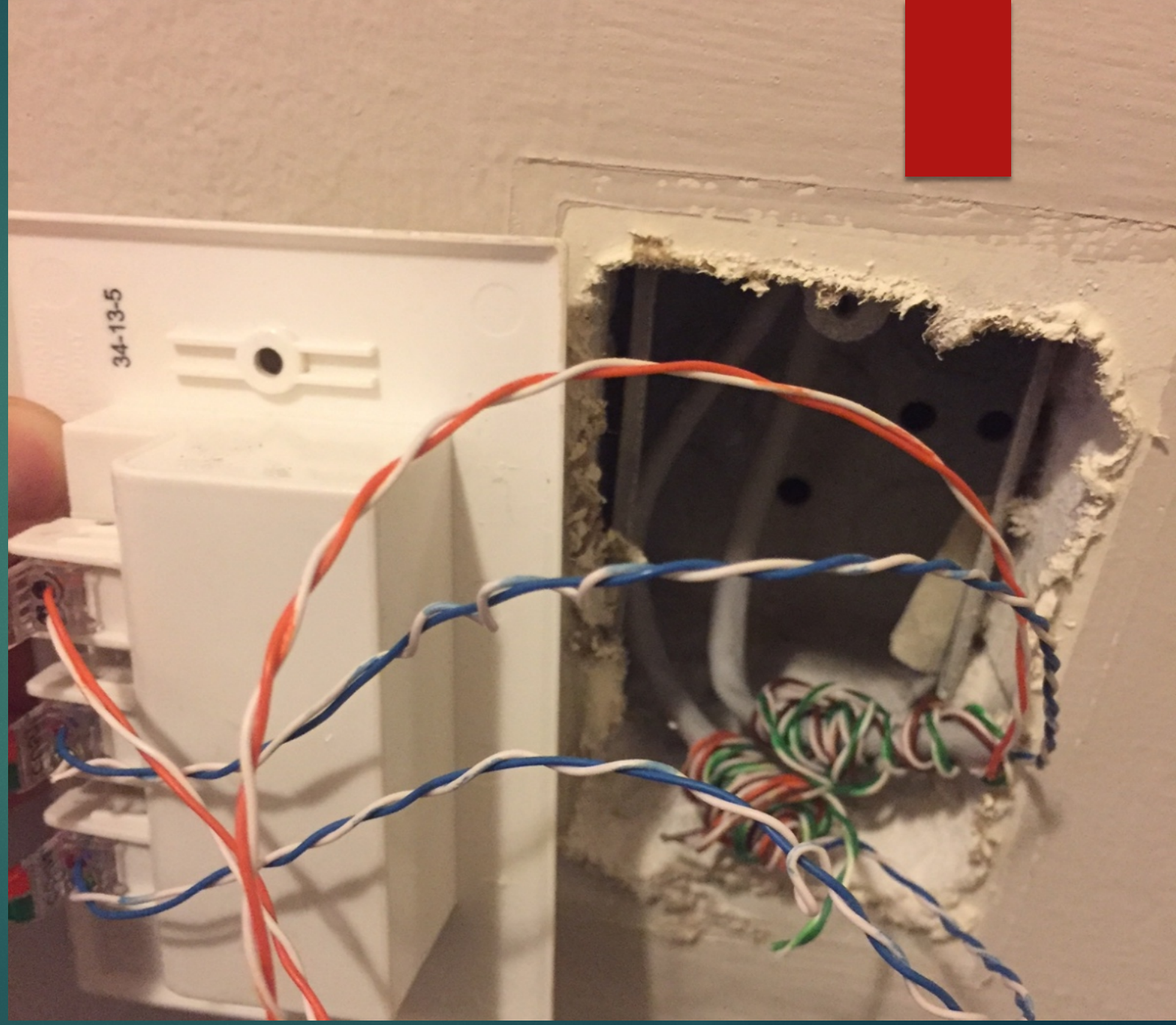
- Initial startup process (Training)
  - CPE and CO detect each other after connection
  - Settings are negotiated based on support and line conditions (Handshaking)
    - Process depends on what configuration is enabled on CO side, and what is supported on CPE side
    - Also depends on Line conditions, what is optional out of what is enabled?
  - Lines start communicating real data (Showtime)
    - Line can adapt real time to changing circumstances depending on settings

DSL Performance Impactors

- Main performance impactors of DSL include
  - Bad / poorly installed cabling
  - Electrical Impulse Noise
  - Crosstalk (interference from other CPEs or external sources)

Poor Cabling

- Poor cabling can result in serious impact
  - Poor Twist on cable can have additional crosstalk
  - Proximity to other cables / electrical devices such as motors can cause interference (cable is often unshielded)
  - Poor installation to jacks can also cause more cross talk



https://forums.tomshardware.com/threads/dsl-apartment-wiring-connections.2974980/

Crosstalk

- Essentially interference from other devices or transmissions
  - Three main types
    - NEXT (Near End cross talk)
      - Generally bad twist on wires (at termination points)
      - Interference between wires on the same side(such as at the jack)
    - FEXT (Far End cross talk)
      - Generally from other devices also transmitting DSL
      - Coupling between wires in the binder
    - Alien
      - Noise from other stuff (Electrical motors etc)
      - Bad cable runs or misbehaving electronics

How is crosstalk dealt with

- Depends on type
  - FEXT
    - Can be improved by better deployment strategies (keeping all DSL similar)
    - Power back offs
    - Vectoring
  - NEXT and Alien
    - Improve cable runs and fix jacks in customer homes

Impulse Noise

- Bursts of very loud noise
  - Three models
    - REIN (repetitive impulse noise) – Bursts of noise over a regular interval, around 1ms max size
    - PEIN (Prolonged impulse noise) – Long prolonged noise levels
    - SHINE (Single high impulse noise) – One single burst of very high nose, great than 10ms in duration.
  - All types cause packet damage/destruction/loss

How does DSL deal with noise (and why do we care?)

- Two major features methods of dealing with impulse noises
  - Forward Error Correction
  - Retransmission
- Why does it matter to us ?
  - Both these features affect the network performance of a DSL connection, mainly affecting latency (but also bandwidth)

Forward Error correction / INP

- Involves multiple methods of correction and encoding
  - Two major concepts
    - Reed Solomon encoding (redundant data encoding / correction/ and detection)
    - Interleaving of data – reduces chance that one entire frame will be destroyed (more on this in a bit)

Reed Solomon Encoding basics

- Used in many forms of digital data (QR codes, CDs, DVDs, barcodes)
- Very simplified explanation
  - Data is separated into blocks called “symbols”, and encoded with redundant data
  - x Check symbols are added to the data
  - Encoded data is transmitted, damage possibly occurs
  - Encoded data is received, and decoded, check symbols are checked
  - Reed Solomon can detect x errored symbols, correct up to x/2 symbols
- Key take aways
  - Redundant data (lowers overhead)
  - En/decoding on each side (takes time -> increased latency)

Reed Soloman in DSL

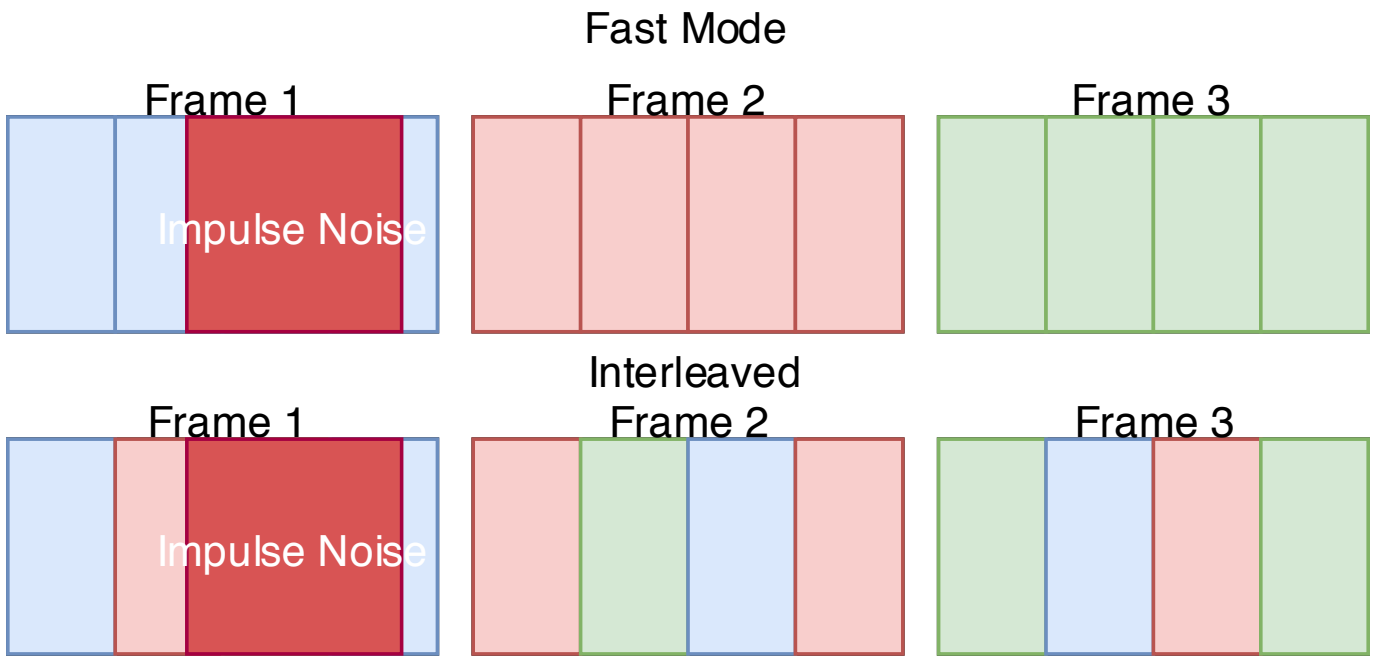
- Level of protection often known as INP (Impulse noise protection)
  - Generally set as a “minimum inp” (INPm)
    - Defined in terms of number of symbols that must be completely repairable regardless of amount of damage
    - Values 0 – 16, with 0 meaning no minimum (fast mode), and 16 meaning 16 symbols
    - Higher the value -> the more redundant data needs to be encoded and the Lower the “goodput” of the line. (lower actual bandwidth as more is used for redundancy)

Interleaving

- Another technique to improve stability and reduce the impact of impulse noise
- Basic idea:
  - Chop data up into many pieces, and send parts of different frames together in one
  - Separates the data out, meaning code words are spread out, and less localized data is likely to be destroyed, and more likely you can correct

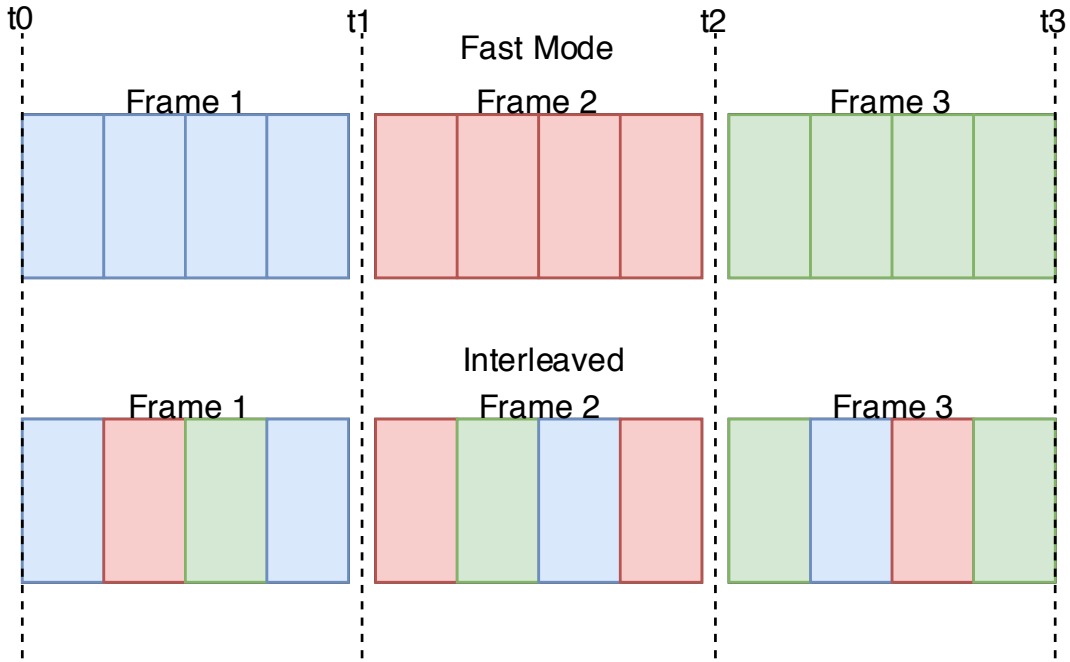
Interleaving 2 (example)

- Contents of one frame now located in 3 different frames
- Impulse noise only destroys part of each frame
- Those parts can be repaired, where as a whole frame may not have been
- Interleaving depth of 3



Interleaving 3 (Latency)

- What is the cost of this? (Increased latency!)
- This latency make some services not work properly (such as VoIP)



Interleaving 4 (DSL Settings)

- Typically controlled with the "Maximum interleaving delay" setting.
  - Defines the maximum allowed interleaving delay (one way), in ms
  - Allowed range 2 – 63ms, typical settings include 8ms/16ms or less.
  - At train-up, the CPE and DSLAM decide what level of interleaving is appropriate – Actual INP is often less than the maximum

A word on fast mode

- Fast mode is operation without interleaving and no impulse noise protection
- Lowest possible latency and highest bandwidth, however high sensitivity to noise!
  - In this mode one way delay may not exceed 2ms
- May be necessary for some services such as VoIP

What is common?

- Deploying both interleaving and encoding (FEC) is very common for DSL lines
  - Most lines need some form of noise protection
  - As loops get longer, lines will often see closer to that maximum interleaving delay
- This means many lines don't run as fast as possible! But require this for stability
- Latency of these lines is tightly lower bounded at the actual interleaving delay, no frame will transmit faster than that.
  - Meaning many lines have minimum latency of 5-16ms just in the last mile.

Another way

- Retransmission
  - Instead of protecting a line by interleaving and encoding additional data, buffer packets and ack / retransmit quickly.
  - This exists for newer VDSL chipsets, and is controllable similarly

Retransmission

- Under Retransmission, data units are know as DTUs (data transmission units)
  - Each DTU receives a frame check sequence, if the DTU is dropped, or the FCS fails, retransmission will be initiated
- Pros
  - Under non-noisy cases, essentially fast mode!
    - Meaning higher throughput and low latency
- Cons
  - Under noisy cases, degraded performance and very long latency
  - Uses memory on devices to buffer

Retransmission vs FEC + Interleaving

- Each has it's own benefit
  - If a line see frequent, short noises. FEC + interleaving can result in a favorable situation. Each noise is corrected and latency cost is paid upfront.
  - If a line sees infrequent, loud noises. Retransmission means that all the times that there is no noise, performance is as good as possible, and bad times will be protected.

How can we perform measurements?

- Need to measure the latency of packets passing through a DSL connection
  - Plan:
    - Use the Spirent Test Center to generate and measure traffic!
    - Use standard DSL equipment (Broadcom chipsets, commercially available products)
    - Use Standard profiles from Broadband Forum
    - Use Standard traffic (iMix)
  - Spirent will provide network performance metrics
  - DSLAM can provide DSL performance metrics

What can the STC do to measure latency

- Best the Spirent can do is Histogram mode:
  - In this case you can define a histogram for latency, measured packets are placed into one of 16 buckets defined by a user
  - Bucket sizes can configured individually for upstream and downstream
- How does Spirent measure latency?
  - Sequence number and timestamp placed into the packet's payload
  - Time measured one directionally from Spirent interface to Spirent interface

DSL Plan

- Measure DSL under a variety of scenarios including:
  - Various loop length
  - With / Without FEC
  - With / Without Retransmission
  - Under impulse noise events
  - Various traffic levels (50% / 90%) of rated maximums
- For future study
  - Various levels of FEC / Interleaving
  - Different Retransmission parameters

NetEm – How can we use it?

- First we need to determine what NetEm can do!
  - Works by shaping a Linux machine ethernet interface
- Plan is to use NetEm on two interfaces, and bridge them together to shape the traffic passing through the machine, giving it upstream / downstream characteristics similar to DSL

How does NetEm Work

- NetEm has the ability to:
  - Impose delay on packets (latency)
    - Fixed delay + Jitter
    - Delay according to a distribution (normal, pareto, paretonormal, or custom)
  - Set maximums on the Bandwidth
  - Packet loss and corruption
- Typical NetEm command:
  - tc qdisc add dev eth0 root netem delay 100ms
    - This sets a fix delay on 100ms, each packet coming through, will have a latency of 100ms

More NetEm

- tc qdisc add dev eth0 root netem delay 100ms 20ms
  - You can additionally specify some jitter, in this case 20ms, packets will range between 80ms and 120ms.
- tc qdisc add dev eth0 root netem rate 100kbit
  - Rate can be easily limited as well via the rate command
- tc qdisc add dev eth0 root netem delay loss 25%
  - 25% of packets would be lost
- But how do we make this match the DSL?

NetEm custom distributions

- One final command for delay!
  - sudo tc qdisc change dev enp3s0f0 root handle 1:0 netem delay mean STD correlation% distribution <filename>
    - Instead of using one of the pre-defined distributions, use a custom file!
  - NetEm contains a tool for creation of these files from your own data,
    - Give it a file containing latency values, and it can generate a distribution file from your data

Our NetEm Plan

- Take DSL Latency measurements under a number of cases
- Create custom distribution files for NetEm to match the latency distribution
  - Bandwidth and other values less of a concern (easy to match)
- How do we convert out measurements from the Spirent into distribution files?

Spirent to NetEm

- Plan is very simple, given data in a histogram, simply place that number of values in a file, and call NetEms table command!
  - Given measurements of 10ms,11ms,12ms,13ms, NetEm wants a file containing 10,11,12,13 (all vertical)
  - We have a histogram with buckets : 0-1ms : 32 | 1-2ms : 45 etc
    - We place 32 0.5ms values, and 45 1.5ms values (using averages)
    - We then feed NetEm the average and standard deviation
  - We see how all this works!

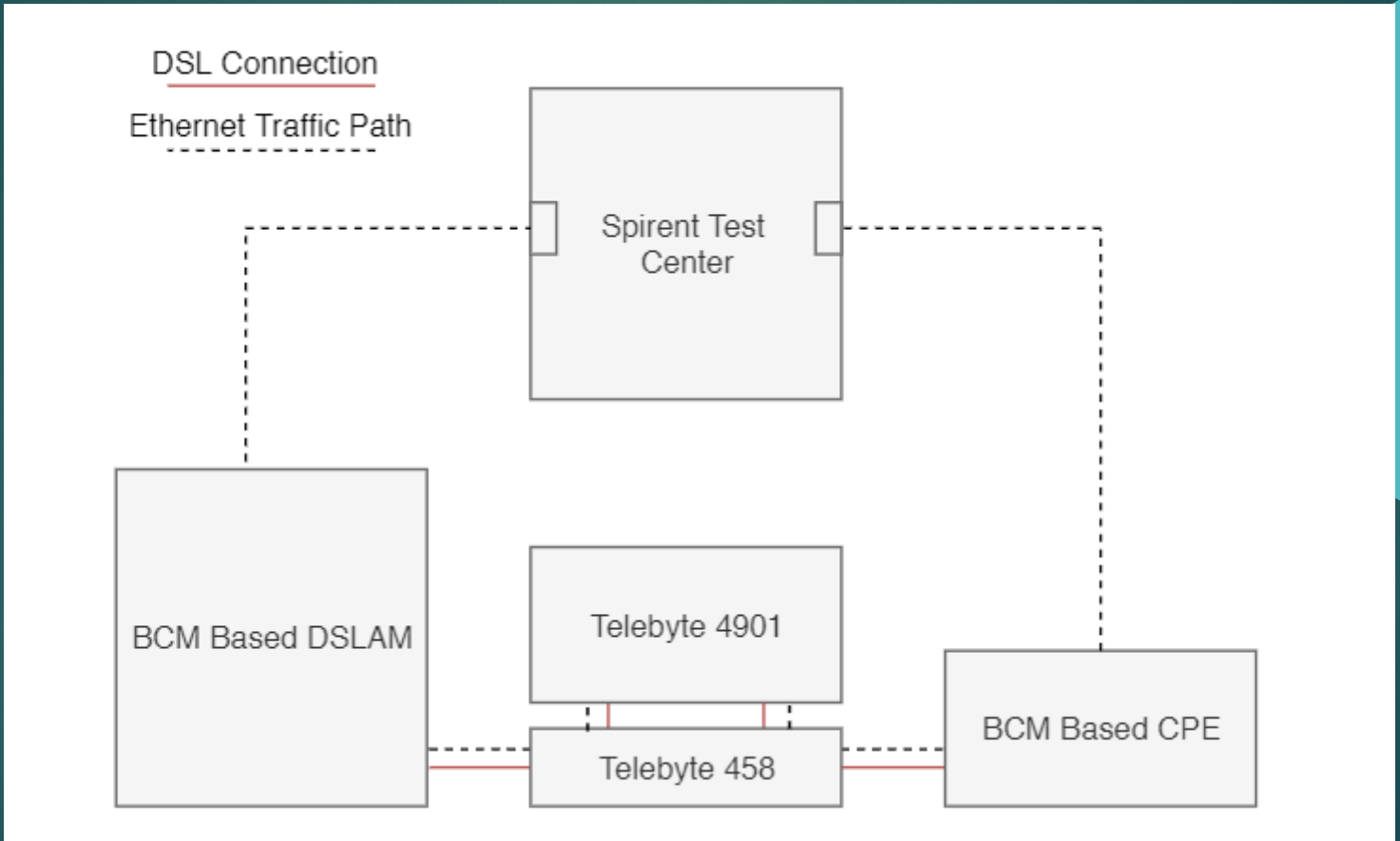
Final Plan

- Measure and study DSL over a multitude of different Scenarios
- Create NetEm latency models of these cases
- Compare them to the actual measurements

The Experiments on DSL

- The basics:
  - Decided on one profile (TR114_ AA8d_AU_RA_I_096_056) from TR114
    - This is a typical 8Mhz profile built from average settings, by default uses 8/8ms max interleaving delay, and 3 min INP
    - Retransmission profile would be added on to this using R-17/2/41 settings from TR-249 (this represents average settings for Retransmission)
  - Broadcom based CPE – commercially available and on latest firmware
  - Broadcom based CO – Also on latest firmware
  - Test equipment all Telebyte
    - 4901 Noise generator
    - 458 Cable simulator

Experimental setup



Traffic Used

- Spirent standard iMix

IP Total Length (Bytes)	Default Ethernet (Bytes)	Weight	Percentage
48	66	7	57.33%
576	594	4	33.33%
1500	1518	1	8.33%

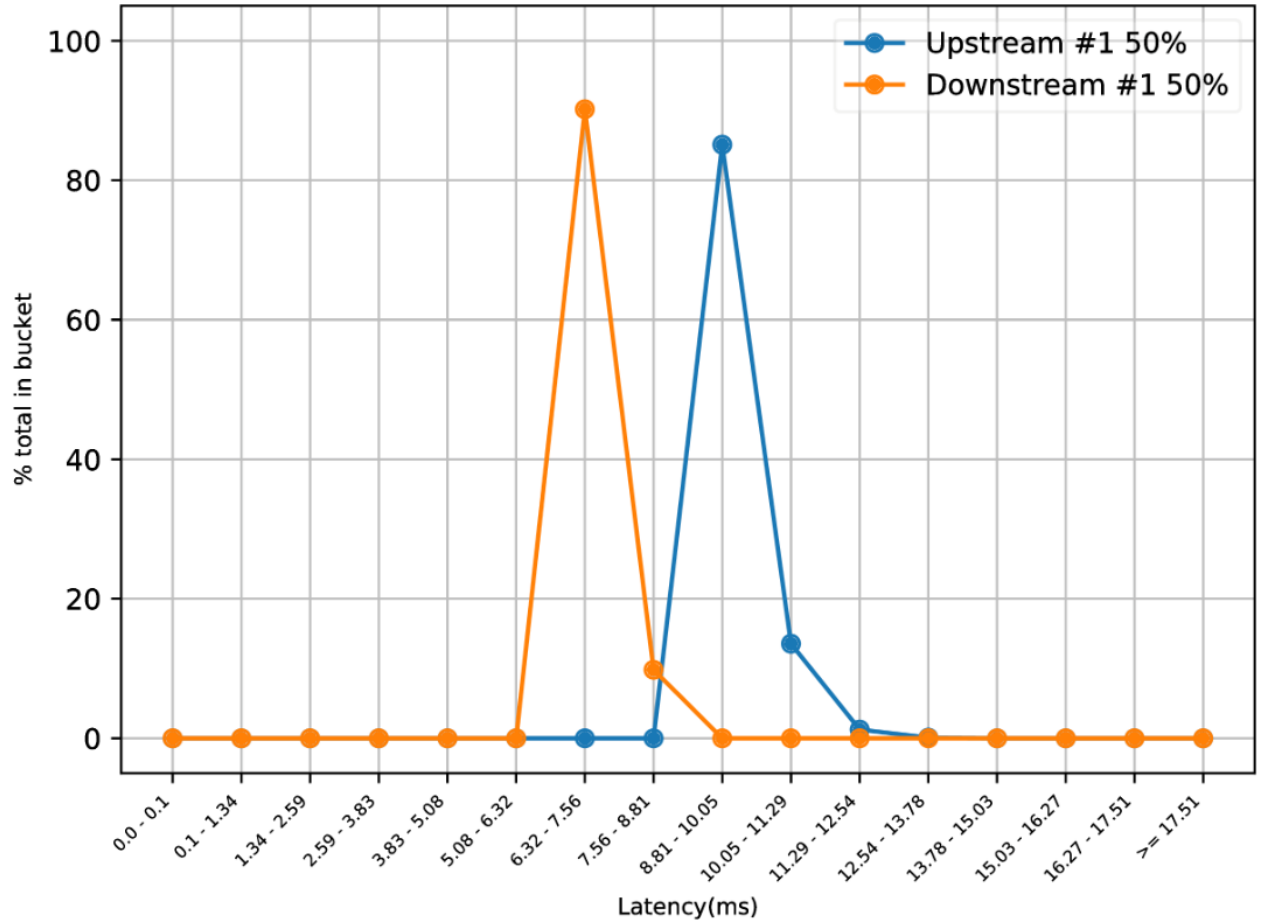
Table 3.1: Spirent IMix traffic distribution

Parallels to test on

- Traffic level
  - 50% traffic, 90% traffic
- Loop length
  - Short loops
  - Long Loops
- Configuration
  - Retransmission
  - FEC + Interleaving
- Impulse Noise
  - Various levels of REIN noise 100us, 1ms.

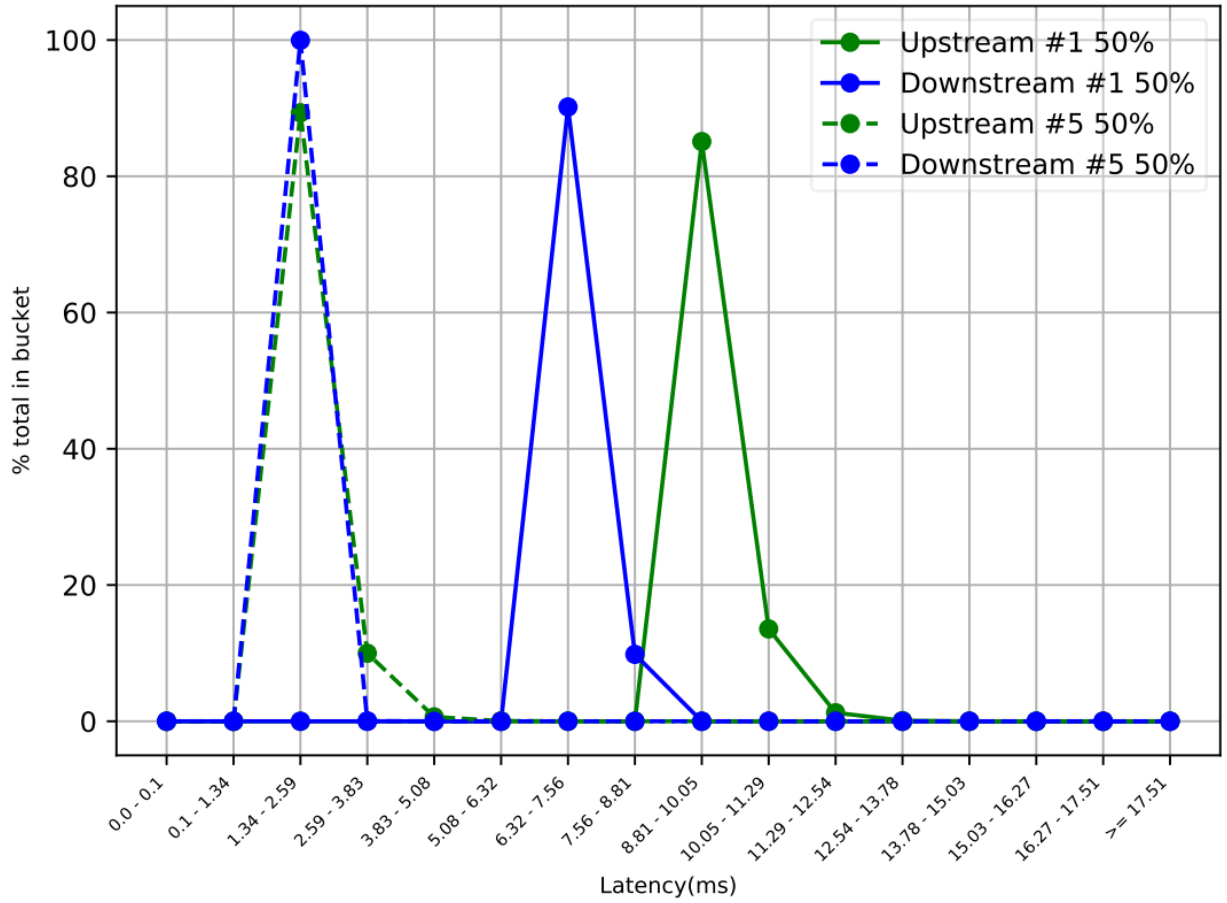
Basics – FEC 50% traffic

- Difference between US and DS – differing actual delay
- Jitter is very low with short loop and no noise
- Minimum is slightly more than the actual interleaving delay



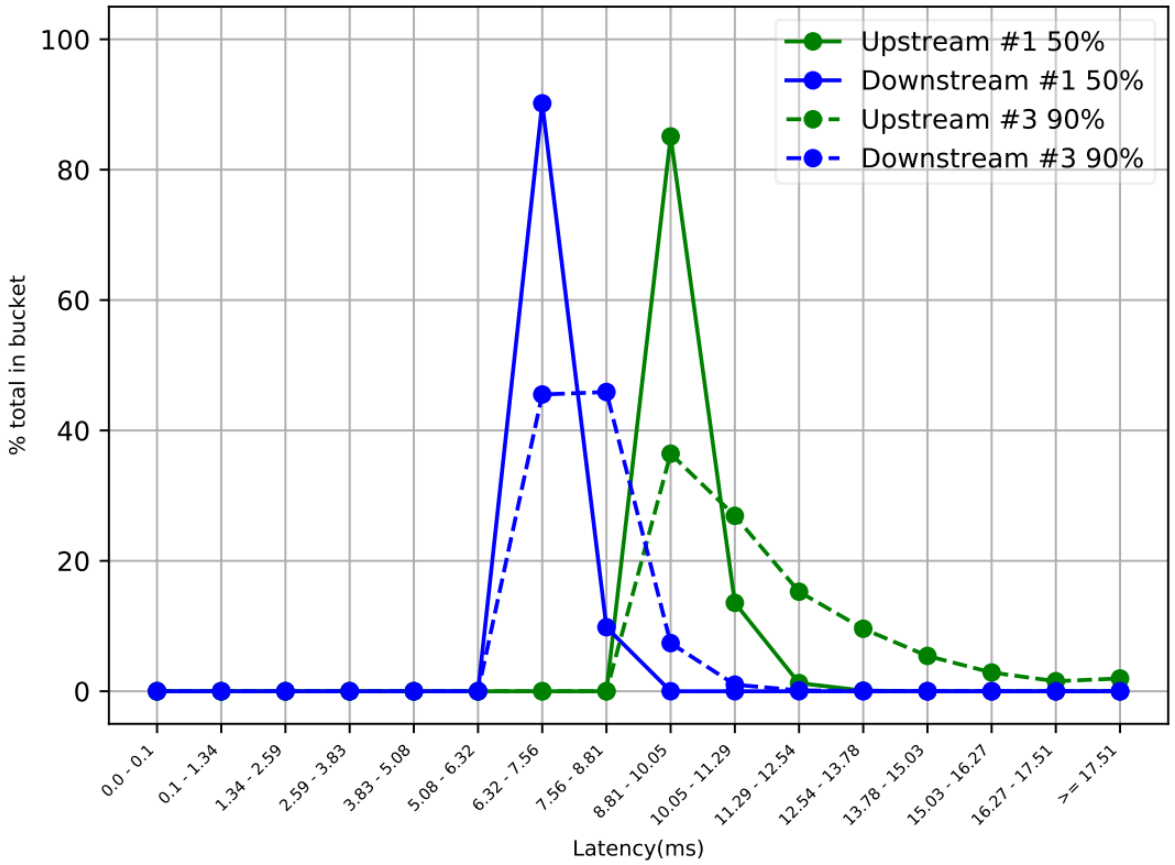
Basics – Retransmission 50% traffic

- Compared to FEC, lower overall latency (Expected)
- Again tightly located with little variation



50% traffic vs 90% traffic FEC

- When compared with 50% traffic, longer tails are seen
- More variation in packet latency
- Much higher maximum
- Consider full queues causing long delays as device becomes fully loaded.
- Upstream sees more variability the downstream.
- Story is similar for Retransmission.

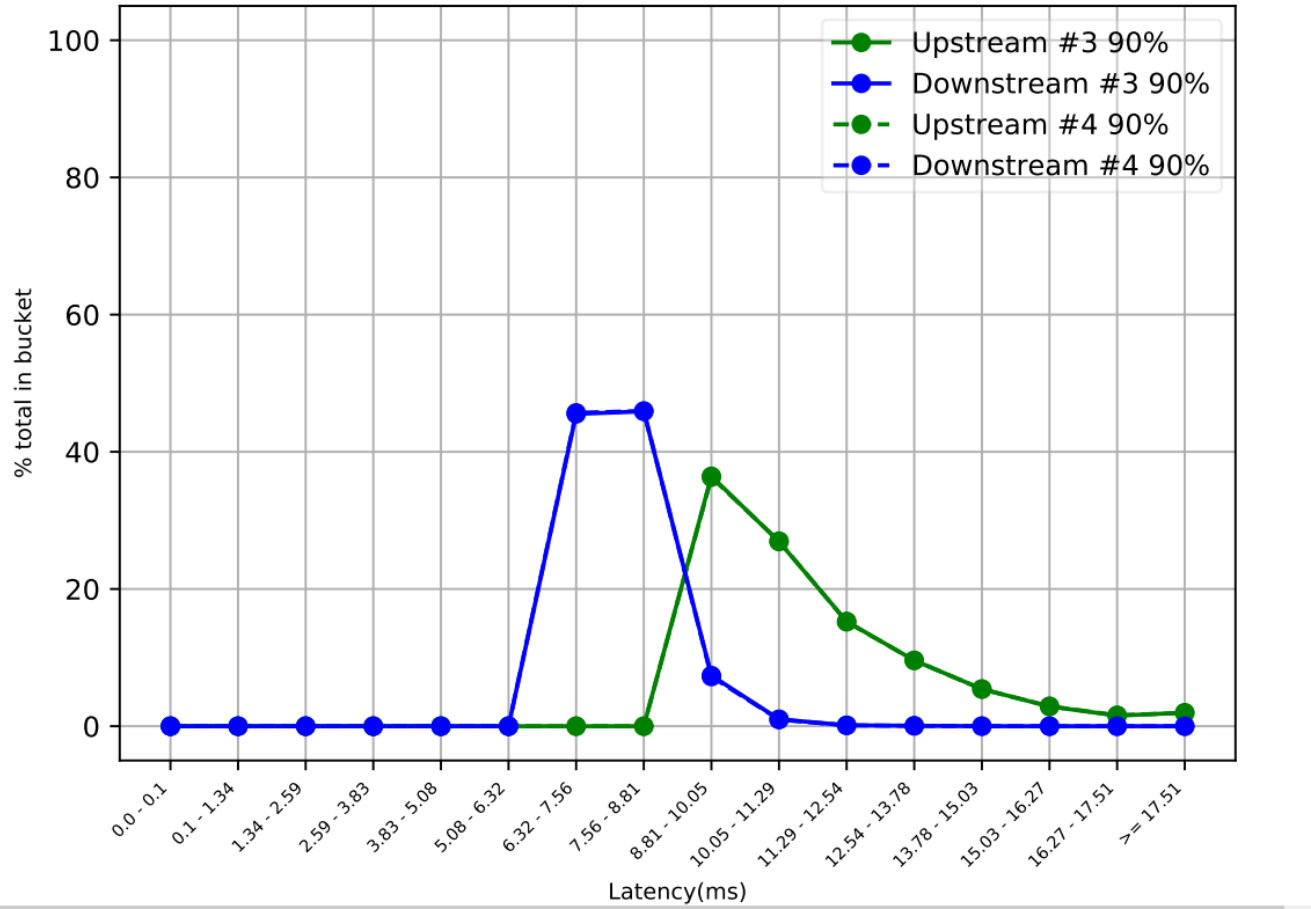


Varying Noise levels

- 100us and 1ms REIN events were tried against both FEC and Retransmission
- FEC was found to not survive the 1ms REIN (possible it would with high INP)
- Retransmission did survive but at the cost of latency

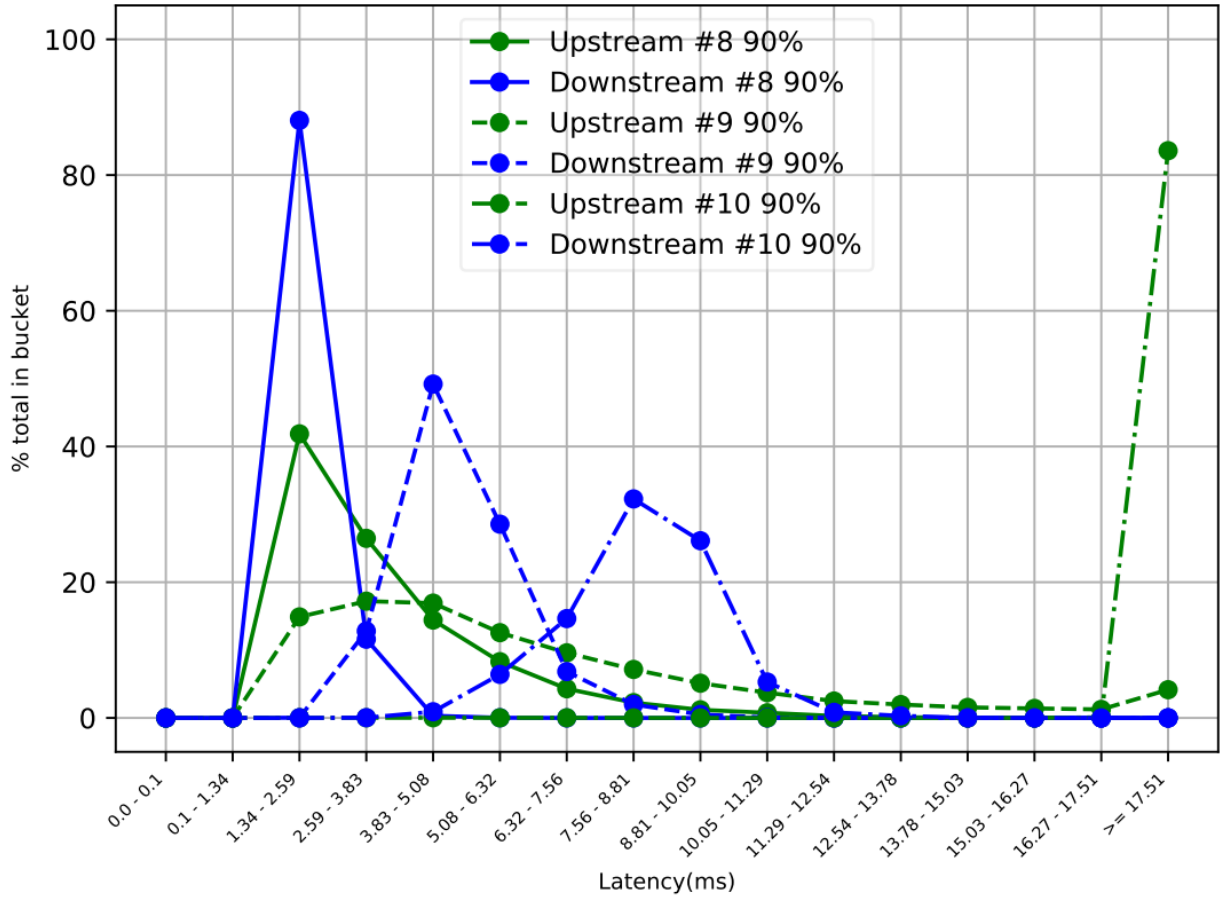
FEC vs REIN

- Well… the latency is the same!
- All the correction is done ahead of time, so all damaged packets are repaired at no significant cost (Cost paid upfront)



Retransmission vs REIN

- Retransmission is a different story.
- Dramatic increase in latency, especially under heaviest REIN condition.
- Correction happens as the noise hits, that's when the cost is paid.
- 8 = Control, 9 = 100us, 10 = 1ms REIN

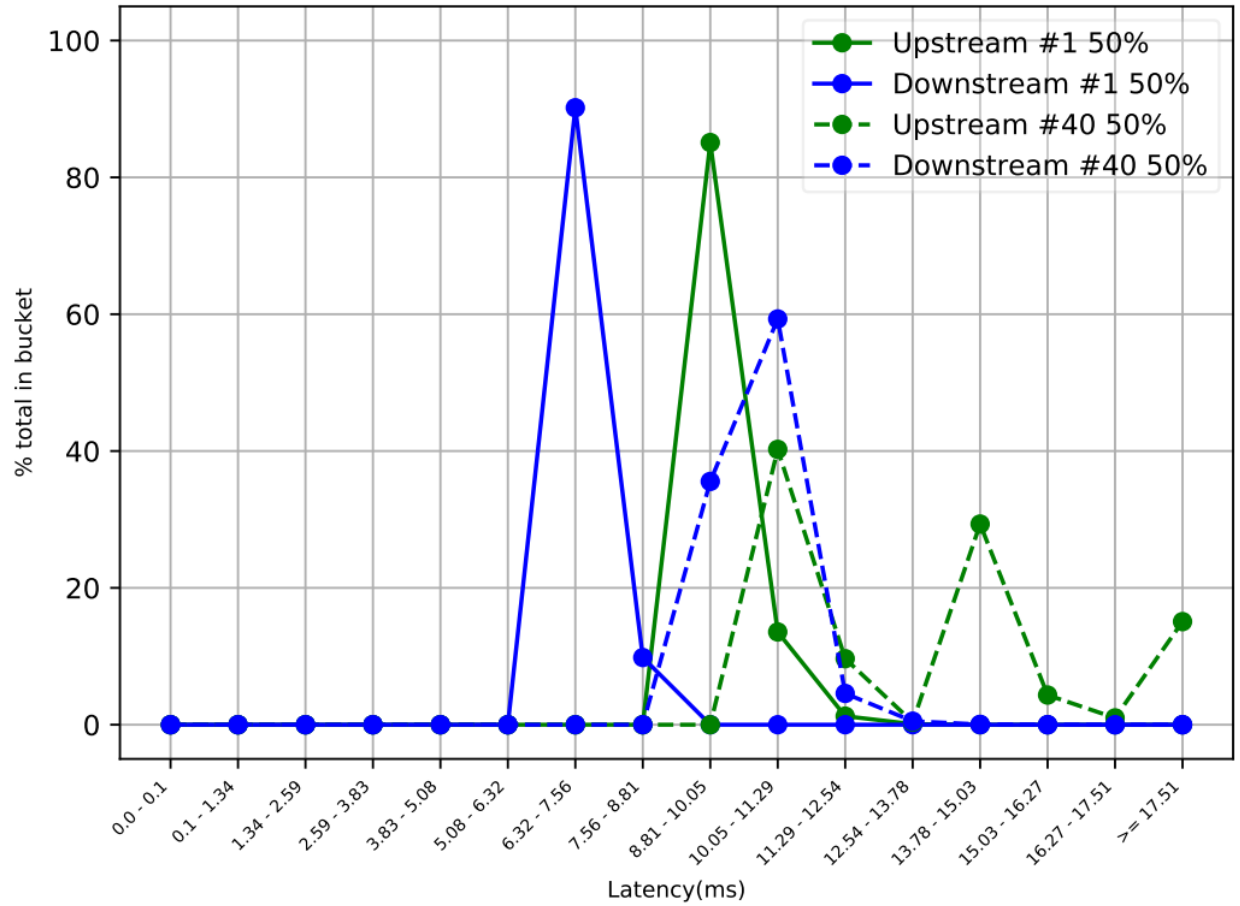


Long Loops

- Ran testing against longer loops, 5350ft, on all conditions. This resulted in approximately 15Mbps downstream rates. The intention was to represent an average customer.

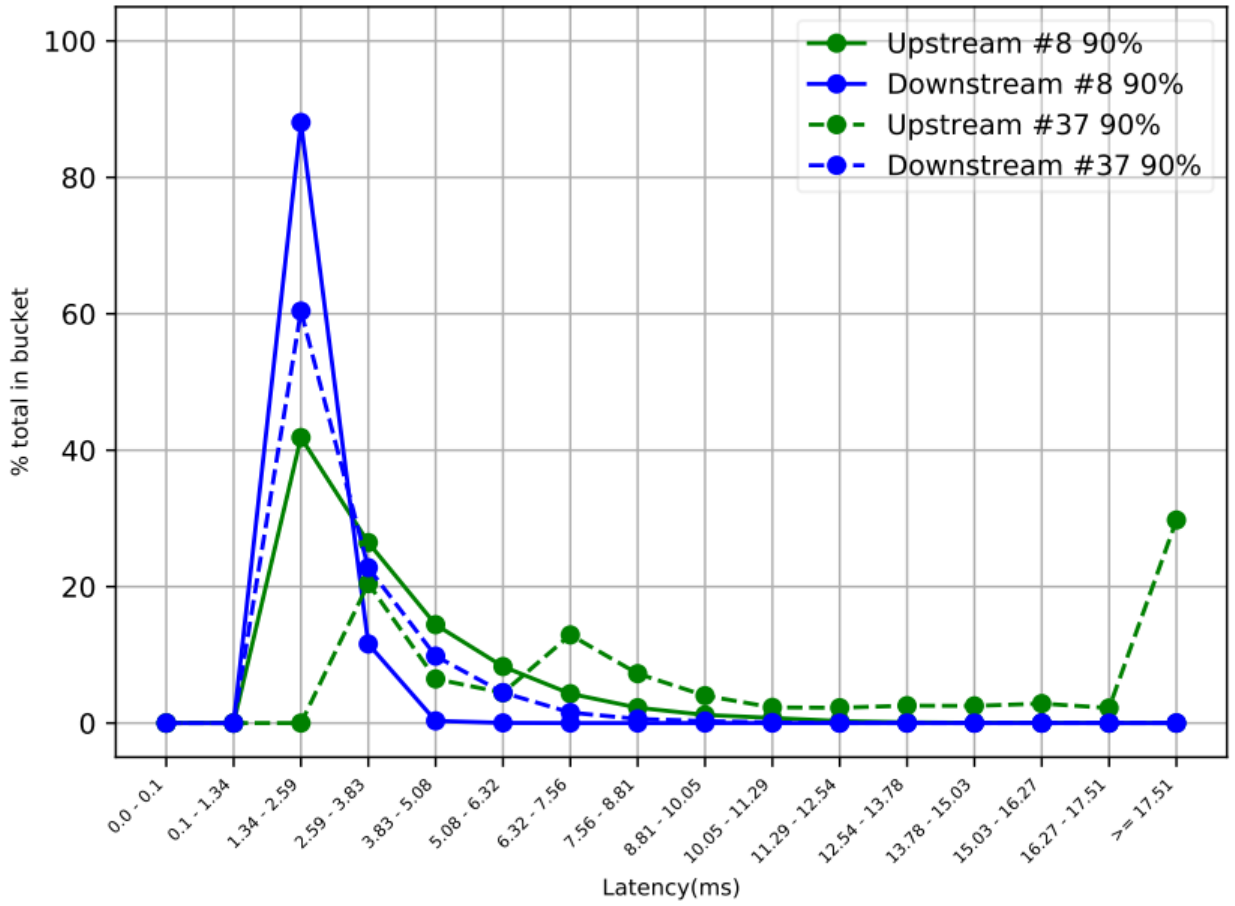
FEC + Interleaving

- 50% traffic, 0ft vs 5350ft loop (1 vs 2)
- Bi-modality in the upstream
- Some very high latency values



Retransmission

- 90% traffic, 0ft vs 5350ft
- Similar upstream bi-modality
- Very long latency packets



Possible issues with data

- Presence of outlier packets with very long latency seems inconsistent (not present in all captures)
  - Possibly related to Packet size
  - Needs more investigation across additional variables (different brand chipsets / CO side implementations / various traffic types)
  - This remains for future study
  - Presence of high amounts of outliers outlines an interesting issue with the NetEm implementation

NetEm models

- To create the models, a simple script was written to turn histogram bucket values into input to the NetEm table maker
  - This script followed our suggested algorithm of taking the average value of each bucket and placing that in the file
  - The script also calculated the mean and standard deviation of the data to use as input to NetEm model.

NetEm recipe

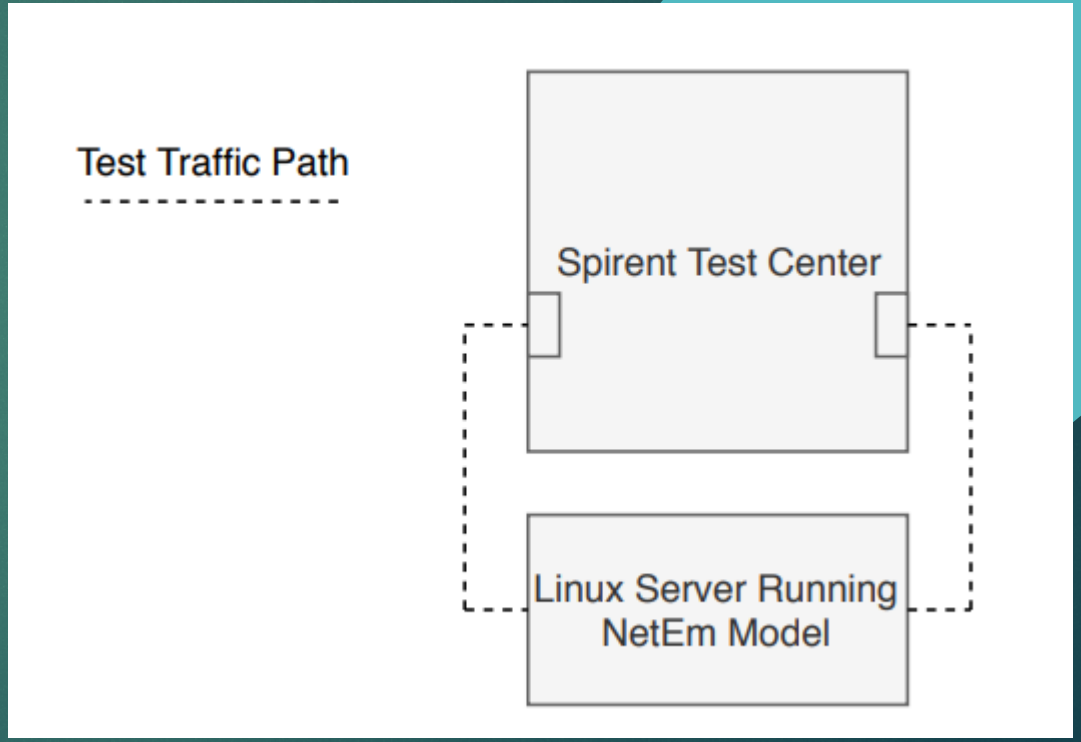
1. sudo tc qdisc change dev enp3s0f0 root handle 1:0 netem delay 7062us 370us 0\% distribution no_rtx_control_1DOWN
2. 2. sudo tc qdisc change dev enp3s0f1 root handle 1:0 netem delay 9631us 509us 0\% distribution no_rtx_control_1UP
3. 3. sudo tc qdisc change dev enp3s0f0 parent 1:1 pfifo limit 1000
4. 4. sudo tc qdisc change dev enp3s0f1 parent 1:1 pfifo limit 1000

Command Breakdown

- sudo tc qdisc change dev enp3s0f0 root handle 1:0 netem delay 7062us 370us 0\% distribution no_rtx_control_1DOWN
  - Applied to interface enp3s0f0
  - Delay with mean of 7062us, and STD of 370us
  - 0% correlation – experimentally found to not help
  - distribution "no_rtx_control_1DOWN" – Custom distribution file to shape according to do
- sudo tc qdisc change dev enp3s0f0 parent 1:1 pfifo limit 1000
  - Telling adapter to use pfifo queuing with a limit of 1000 packets in the queue

NetEm Models

- These models were made for every test case then tested with the initial traffic used in the DSL measurement, the results were then compared.
- Experimental setup ->

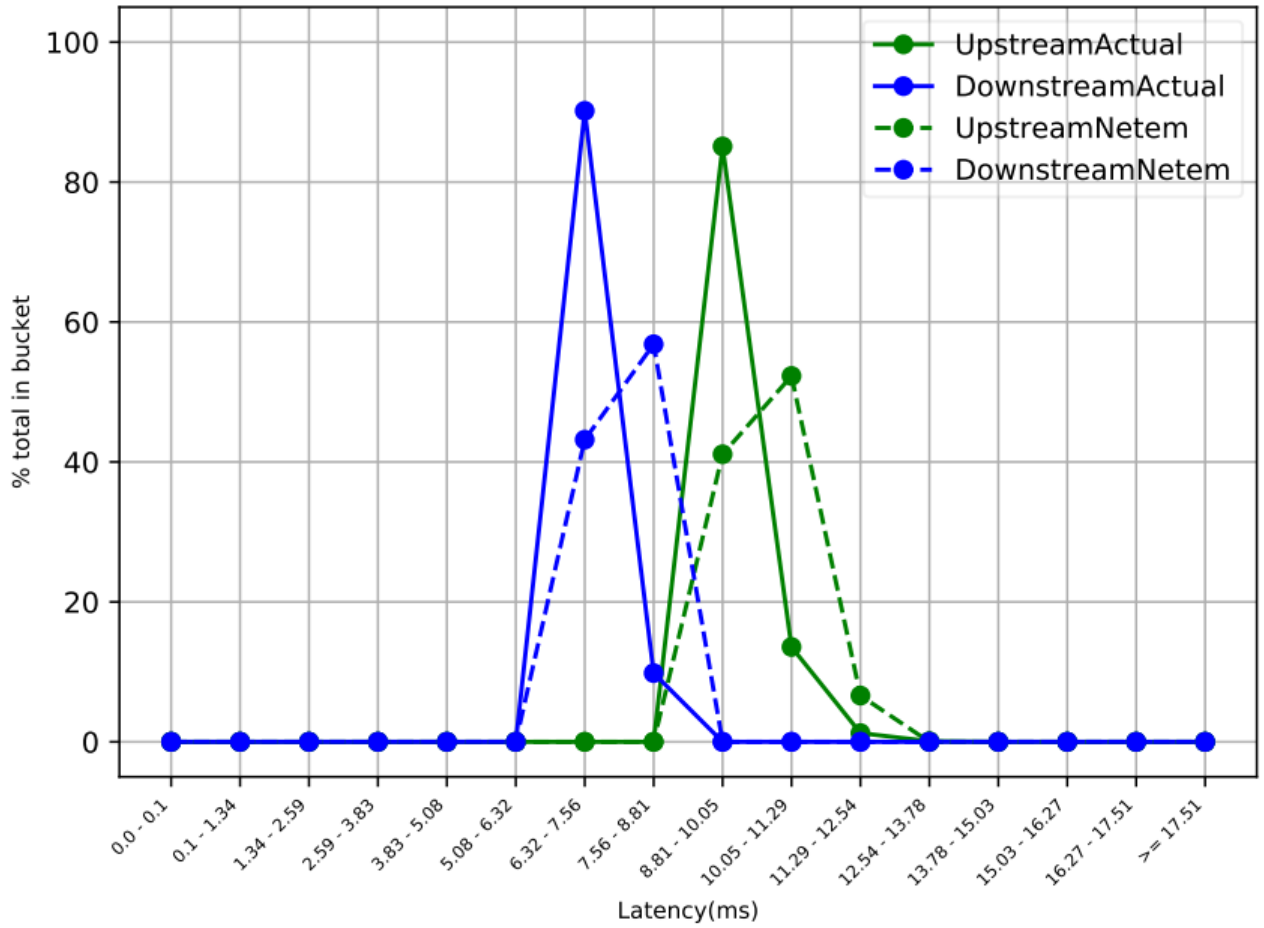


Comparisons!

- Again Compared against the same cases
  - Traffic level
    - 50% traffic, 90% traffic
  - Loop length
    - Short loops
    - Long Loops
  - Configuration
    - Retransmission
    - FEC + Interleaving
  - Impulse Noise
    - Various levels of REIN noise 100us, 1ms.

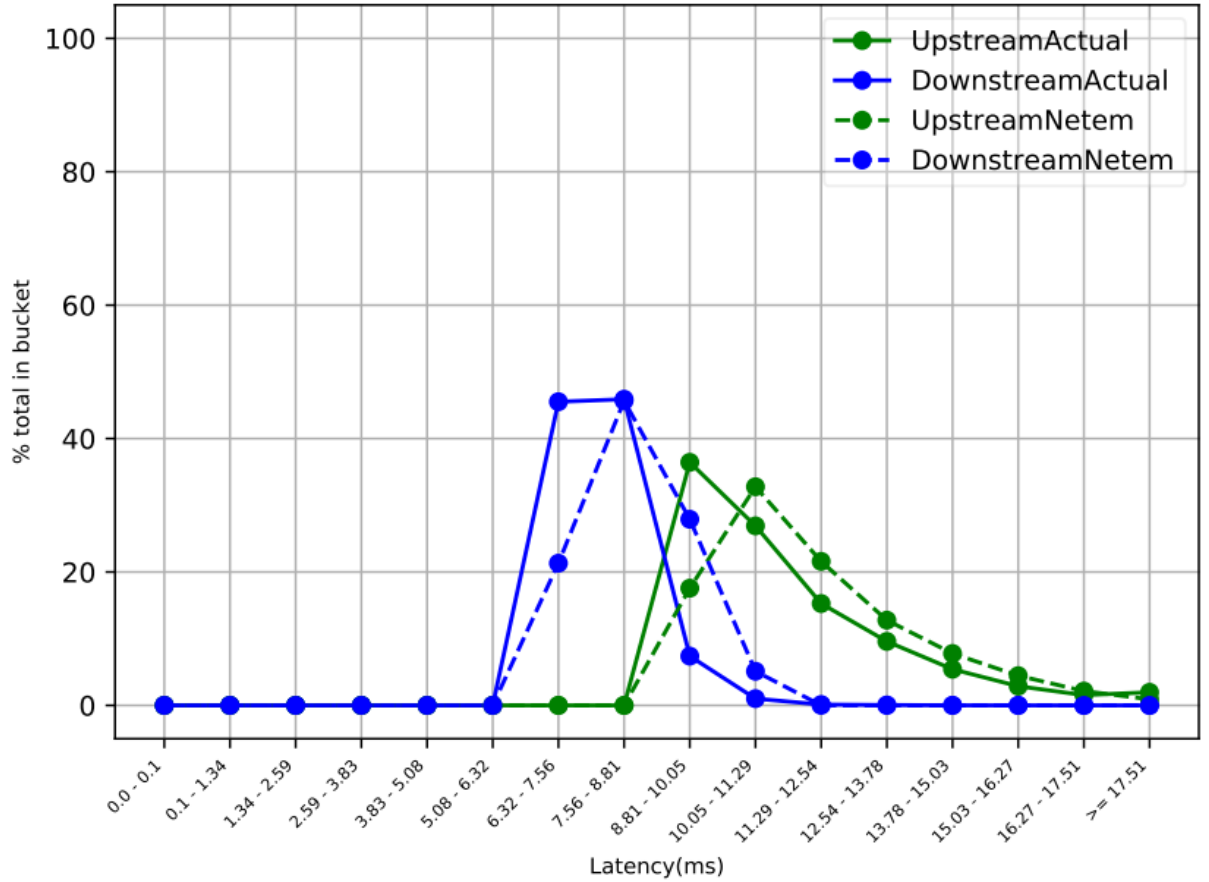
Basic FEC + Interleaving

- Match is fairly good!
- Slight skew toward higher values when compared with reality.
- Good enough for our purposes for sure.



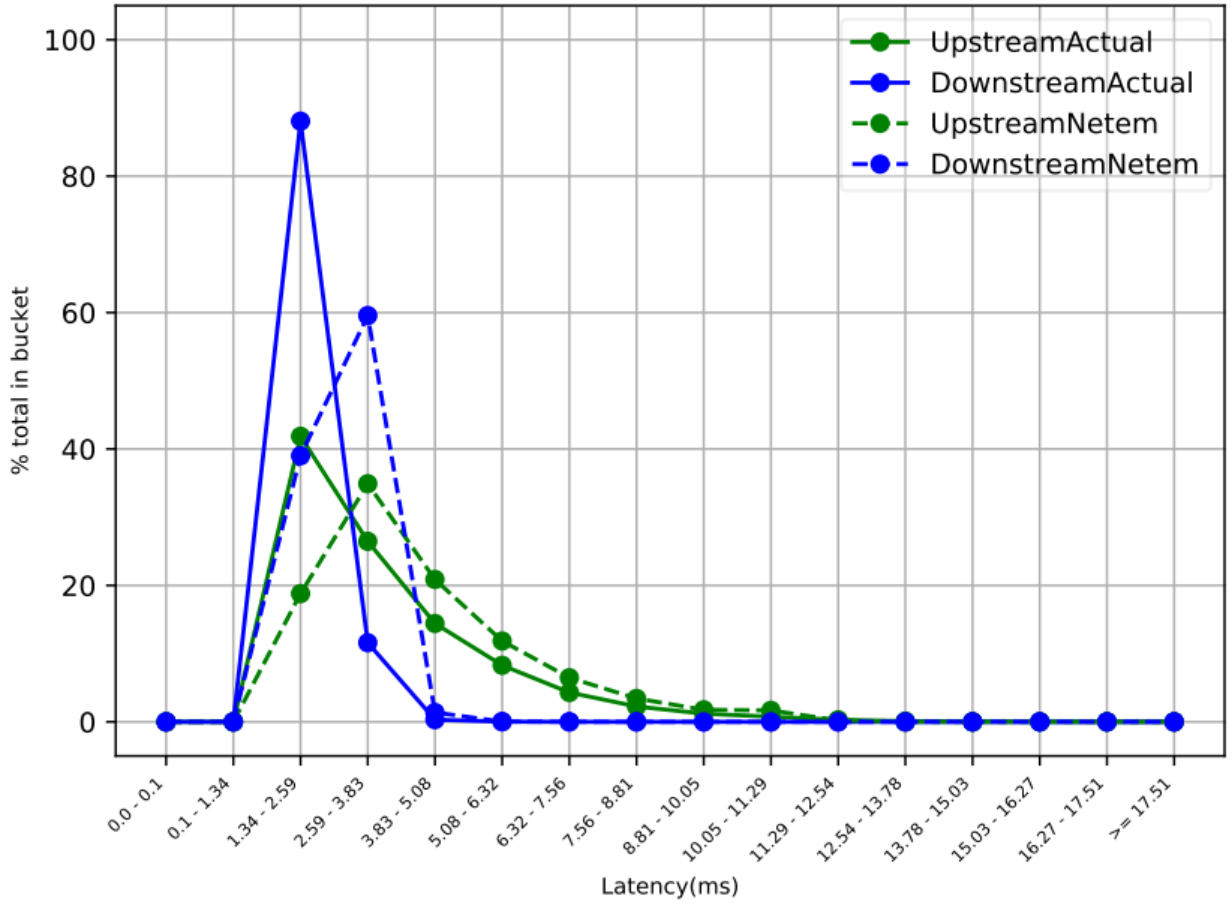
With 90% traffic load

- Again quite good
- Matches both shapes well, still with a skew toward higher values (theme is emerging here).



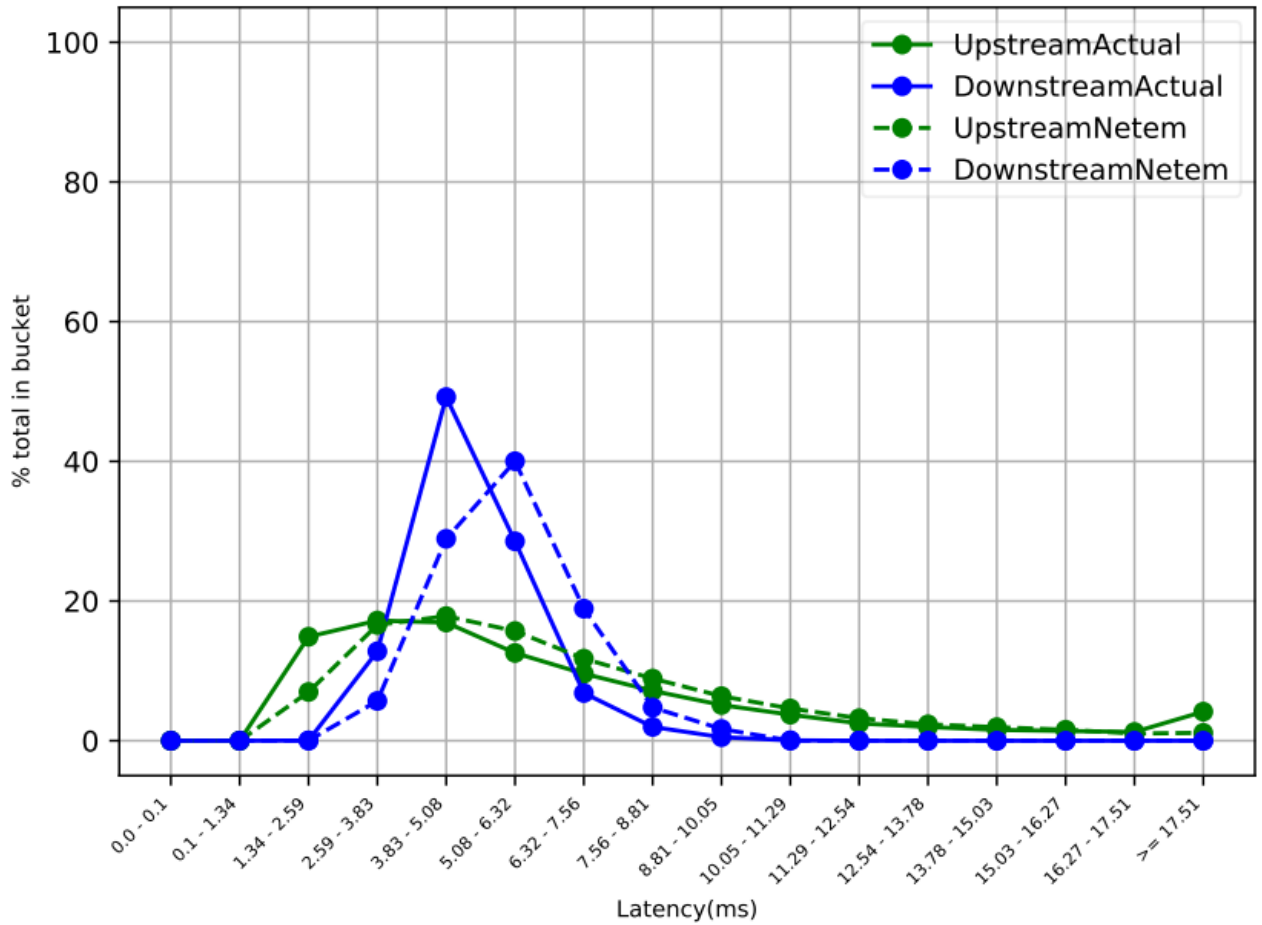
Retransmission 90%

- Similar performance to other cases
  - Good match of long sloping tail.
  - Adequate match of peak
  - Still skewed a bit high.



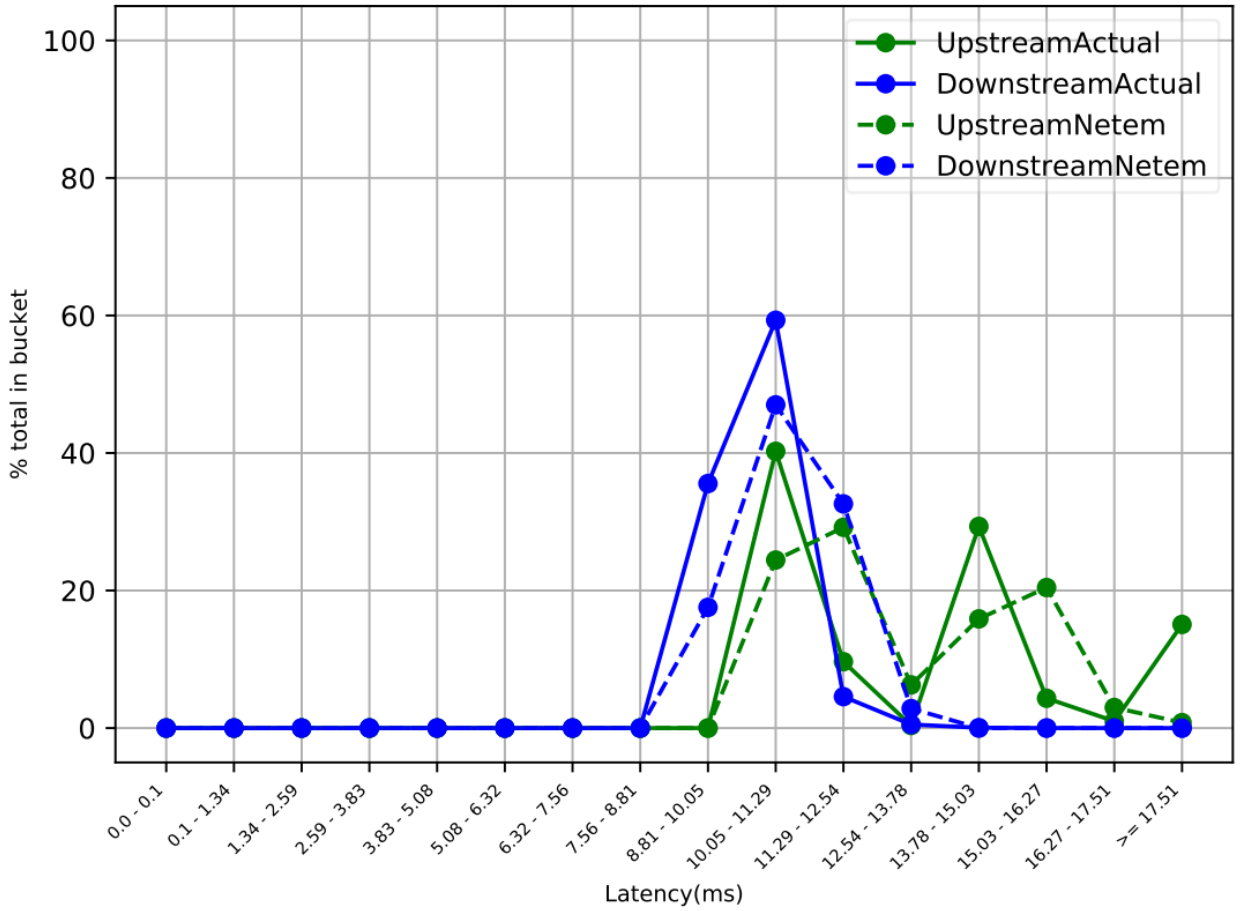
100us REIN Noise Retransmission

- Very good match of long slope in US
- Good match of Downstream peak
- Still slight skew toward higher values
- No match of values in the last bucket, a hint of things to come



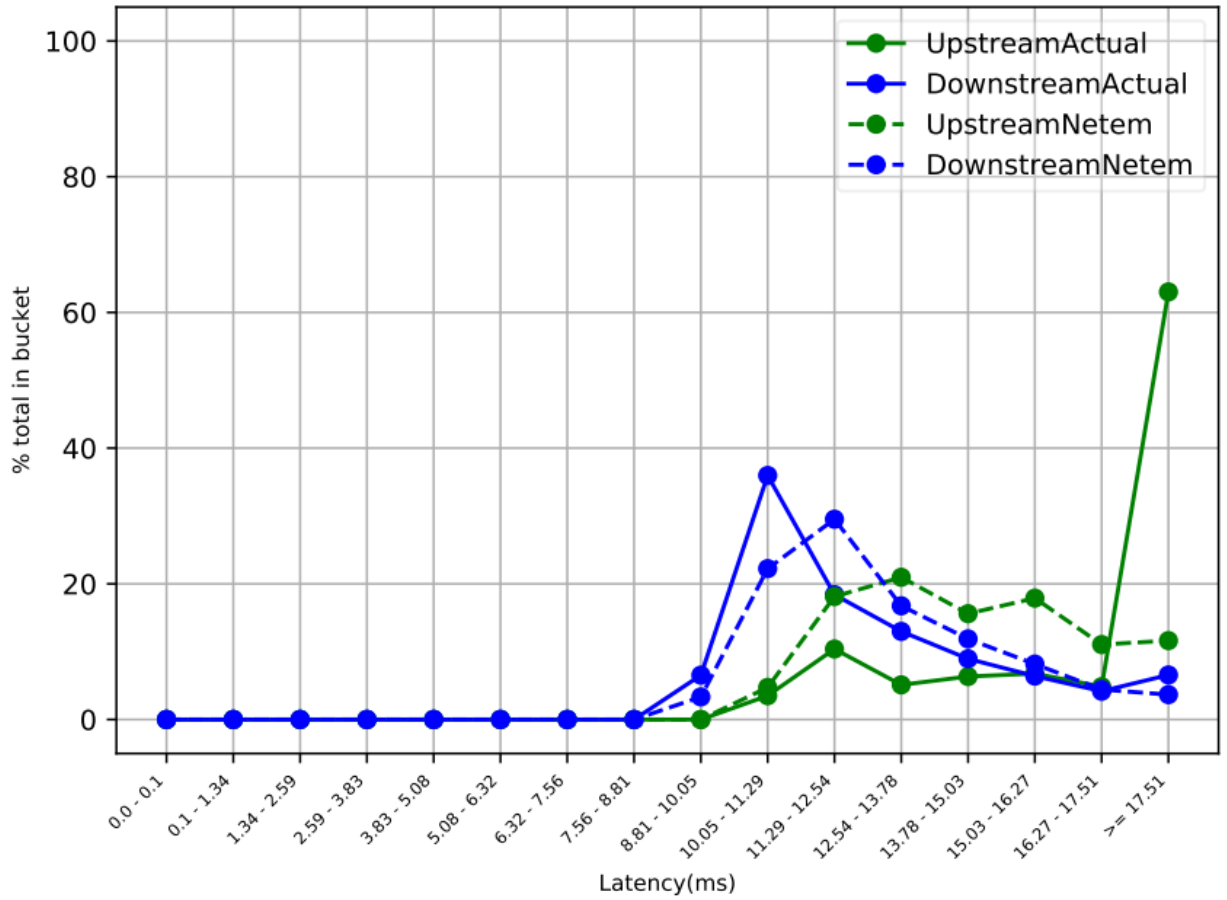
FEC Long Loops (5350 ft) 50% traffic

- Long loops typically see a bi-modal upstream
- Similar Bi-modality was modeled
- Same skew toward higher values
- Values at the end of the measured DSL, were not modeled!



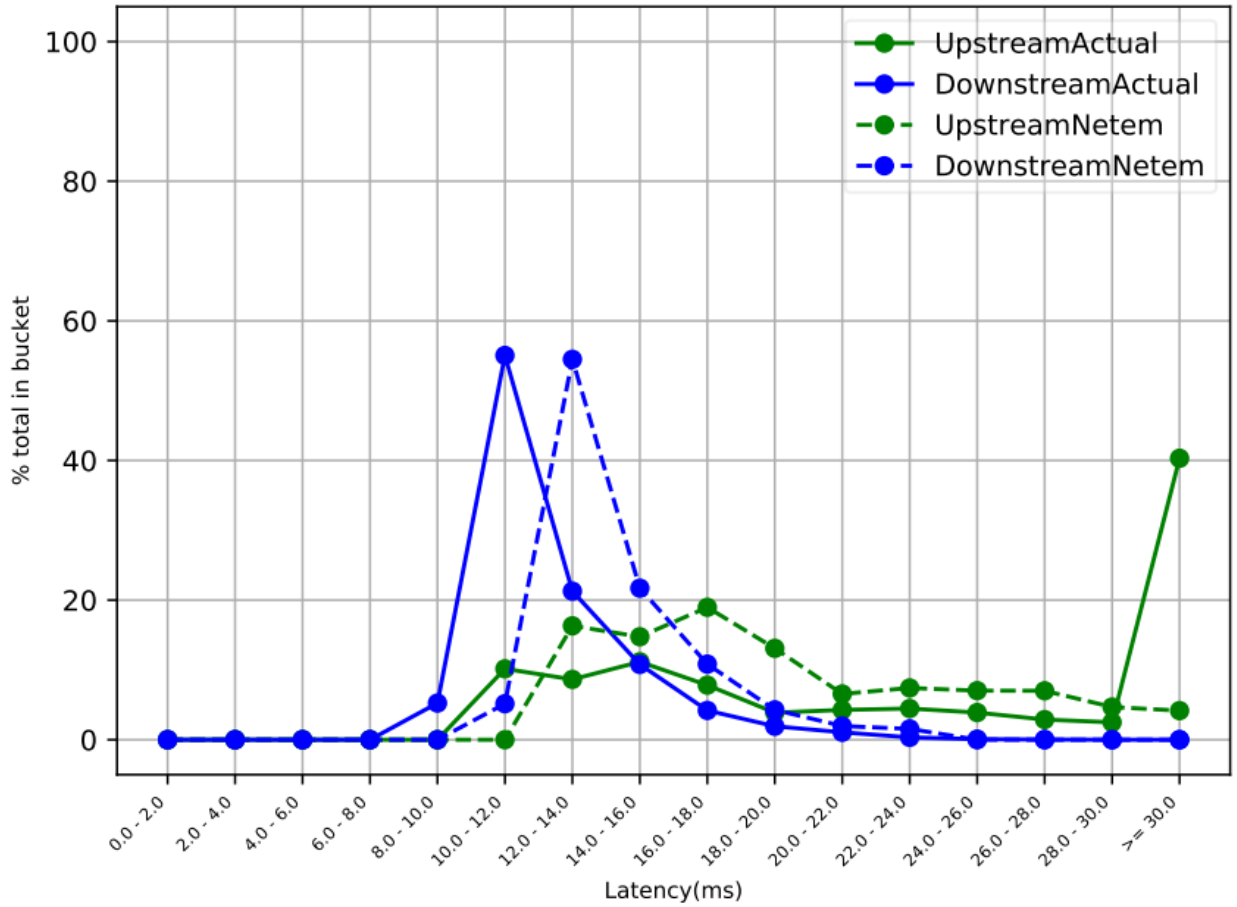
FEC 90% at 5350 loop

- Starts to be pushed off of the graph.
- Seeing lots of very long packets here.
- NetEm does not model packets in the last bucket.



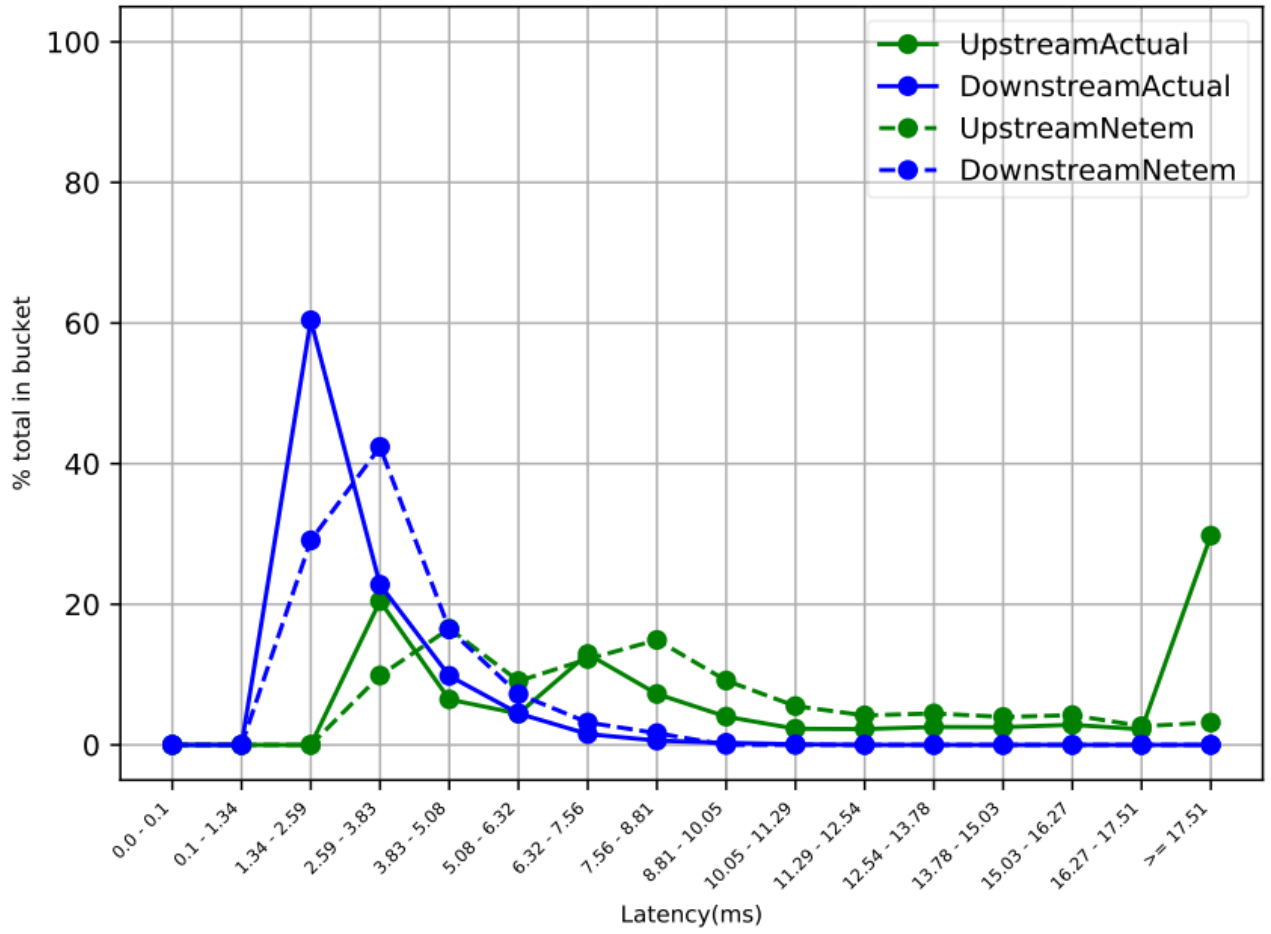
FEC 90% at 5350ft loop (Wider buckets)

- Better match than previous
- Still have values off the end of the graph!
  - Spirent resolution issue.



Retransmission 90% 5350ft loop

- Match is fairly good and captures Bi-modal nature
- Last bucket not modeled.



Model Observations

- Overall, effective for our purposes. Provides a close enough estimation of DSL performance to give a better indication of real performance when compared with simpler models.
- Successful in matching the shape of multiple cases
- If anything, harder cases for devices than reality (modeled latency slighter higher than actual reality)
- Accuracy is enough for testing purposes

Model Issues

- A few main issues
  - Skew toward higher values
  - Resolution issues with highly varied data (losing values in that last bucket)
  - Packet Reordering !

Skew towards higher values

- Two theories
  - NetEm has some inherent latency (around 350us), which will be added to each value
  - Our process of using the average bucket size gives a slight skew toward the higher values and under-represents the lower values (particularly in the last bucket)

Both of these can be addressed in future work.

Resolution issues

- The main limitation of this process is resolution within the Spirent.
  - As we look at a wider range of values in the histogram, we lose resolution between the bucket sizes
    - Wider buckets = less detail
  - Doesn't significantly affect each case, but any highly variable case has issues.

Packet Re-ordering

- In the NetEm testing the majority of the packets become re-ordered
  - Is not present in the DSL (maximum of 10 reordered packets)
  - Could affect higher level protocols
- This is a property of NetEm and would require modifications to the tool to allow a shape to be kept with no-reordering.

Long loop issues

- Initial investigations indicate this may be related to packet size (some size packets move faster than others!)
  - This could possibly be an optimization for certain services
  - Will need to look at other chipsets to see if behavior exists across vendors.

Conclusion

- Our models are good!
  - Useful to model latency performance similar to DSL
  - Much better indicator of reality than simpler models (DSL has asymmetric behavior, including bi-modality)
- Lots of further work to do!
  - Improvements to the models
  - More study of DSL (lots more uninvestigated settings)
  - Possibly find tools with better resolution for measuring latency
  - Test other things over our models!
  - Deep look into long loop data