90% of the internet uses TCP

Multiple paths to the internet at each time 🡪 many redundant paths between servers, more reliability

TCP 🡪 single path protocol for each connection, single process of end device

TCP traffic will only flow between two IP addresses

TCP will need to reconnect if you switch places 🡪 not nice for mobile users

ECMP 🡪 Equal-cost multipath

Challenges with TCP following single path that is why we look at ECMP

Traditional Wireless

You want to use wifi if 3G tower fails and vice versa, but this cannot happen with TCP

Equal Cost Multiipath

5 tuple – source IP, destination IP, source port, destination port and the protocol being used

Path mapped by those 5 elements

Hash(5 tuple) a new flow would have one of the elements inside the hash would be different

New Bytestream Model

Internet is generally multipath

The flow from clients to server should untilize as many of the available paths as possible

Stream of traffic should be able to use all available paths

Design objectives

Applications run over TCP should also be able to run over multipath TCP

Must be able to run over any internet path

Multiple subflows to be combined in a session

TCP Connection Establishment

Win = 4 how many messages the server is prepared to send at this point

Naïve approach

Just send message through both paths. The fist path has established connection, the other one does not. If you just send through both, the path that is not established the message will be stopped in the middle boxes, by the firewall and they will be stopped.

TCP Connection Establishment is different under multipath

MP\_Capable(X) <- capable of using the multipath, will indicate whether it can do multipath

MP\_Join(Y) <- uses new connection to join existing connection

Now the middle boxes will understand that there are two flows

Design Decisions

* A multipath TCP connection is composed of one or more regular, combined TCP subflows
* Hoat maintains state that glues TCP subflows that compose a TCP multipath connection together
* Each TCP subflow is sent over a single path and appears like a regular TCP connection along this path

Multipath TCP

Bytestream service

Only different within transport multiple TCP flows

Flow control

Flow control sent by the server indicating how many bytes it is able to buffer, indicated to sender by window size. How should this be inidicated in multipath

* How should the window-based flow control be performed?
  + Independent windows on each TCP subflow
    - Performance not too good
    - Impossible to retransmit, where window is already full
    - Problem when one window sends and the other one is blocked and it remains blocked and is needed by the other side
  + A single window that is shared among all TCP subflows
    - Window size will be adjusted per messages sent on both sides not depending on the subflow
  + Middle box impact
    - Can reduce window to reduce traffic
    - The receiver will discern the difference and then use the largest window received

Multipath TCP Windows

* Multipath TCP maintains one window per multipath TCP connection
* - Window is relative to the last acked data
* Window is shared among all subflows
* Window is transmitted inside the window field of the regular TCP header
* If middleboxes change window
  + Use largest window received at MPTCP-level. Use received window over each subflow to cope with the flow control imposed by the middle box

Two Subflow Example

* You can add a new subflow, remove an existing subflow. Should act as a single pipe that is consistent

Link Subflows

* Say they are connected, but the problem is that port numbers are not a very realiable source, because NAT boxes could change port numbers
* Next method is to use tokens. Client has a token and server has one too. The second connection can use the server token to indicate to the server which connection to associate it to

Subflow Agility

* Multipath TCP supports
* Addition of subflows
* Removal of subflows

The order could be changed when transferring data over subpaths. They could be rearranged using sequence numbers. BUT middle boxes do not like things that are unpredictable like this. They will not like the variety in flow. GAP sequence, some DPI will not allow this.

To solve this use TWO LEVELS OF SEQUENCE NUMBERS

Top layer is **data sequence number ->**  This is for entire data sequence so we can align the ordering of the two sequences

TCP1 sequence number

TCP 2 sequence number

They can be rearragned on the user side if there is a problem

How to deal with losses

* Data losses over one TCP subflow
  + Fast retransmit and timeout as in regular TCP
* If one subpath fails it will be retransmitted on the other subflow

**Question:** if the subflow is lost on the one path, it will be transmitted on the other path. But will is disregard the lost path or will is try again after a certain amount of time?

It is more on an implementation issue. But if this path would need to be restablished with a new path and new ports and IP addresses.

Retransmissions heuristics

* Fast retransmit is performed on the same subflow as the original transmission
* Upon timeout expiration, reevaluate whether the segment could be retransmitted over another subflow
* Upon loss of a subflow, all the unacknowledged data is retransmitted on other subflows

Week 2 Lecture

Has not been implemented everywhere

Quick is implemented by google. So TCP is not being used as much. So from 95% TCP users to 85%, because many people use chrome.