Multipath Transport on the Internet

Sebastian Siikavirta, Gautam Raj Moktan, Antti Jaakkola

Helsinki University of Technology, Comnet
Finland
Email: sebastian.siikavirta@tkk.fi, gmoktan@cc.hut.fi, atjaakko@cc.hut.fi

Abstract

Modern Internet end hosts like mobile phones have several interfaces to connect to the network but can use only one at a time with current Internet transport protocols. TCP and UDP are the dominant protocols of the transport layer and most firewalls and NATs primarily allow them. They are single link protocols. For the purpose of making multipath transport possible that also works seamlessly with the current Internet, this work devises a mechanism that utilizes existing and experimental standards to provide the intended functionality.

This research focuses on a simple solution that makes use of FEC (forward error correction) coding to avoid retransmissions of packets as in TCP. It also uses DCCP to provide congestion control mechanisms.

Thus, by using a multipath mechanism over DCCP that is tunneled inside UDP, a method is obtained that can aggregate several different connections for better reliability and increased bandwidth compared to a single connection. Multipath transport is built with transmission windows and cumulative acknowledgements of encoded packets.

I. Introduction

Most smart devices these days have several connections to the network but only single connection is used at a time. This limitation is caused by the current transport protocols which are single path only since they were designed for devices having a single network interface. Multipath transport is method to have one logical connection over multiple physical connections. The multipath transport should be transparent for the applications, i.e., no alteration of the application should be required. Benefit is increased reliability and bandwidth compared to any single connection.

This research is focused to take advantage of multiple network connections at the end hosts. It is common that two paths over Internet share common sub path or link. This multipath solution does not increase reliability nor bandwidth aggregation in these links. We assume that these common shared links are core network and thus do not suffer the same level of unreliability as the first and the last hops from the device. The core network rarely is the bottleneck of the bandwidth between the endpoints.

The main reason for the multipath scheme is the increased reliability. Another benefit of multipath transport is the increased bandwidth compared to single link. Multipath aggregation tries to take advantage of all the bandwidth of every link.

II. Working Principle

Transmitted data will be encoded with forward error correction. With forward error corrections it is not required to keep track of lost packets. Senders and receivers do not have to signal which of the packets were successfully transmitted. This also implies that the order of the received packets is irrelevant. The sender will continue to send data as long as the receiver informs that it has received enough data to decode the original. This idea is illustrated at Figure 1.

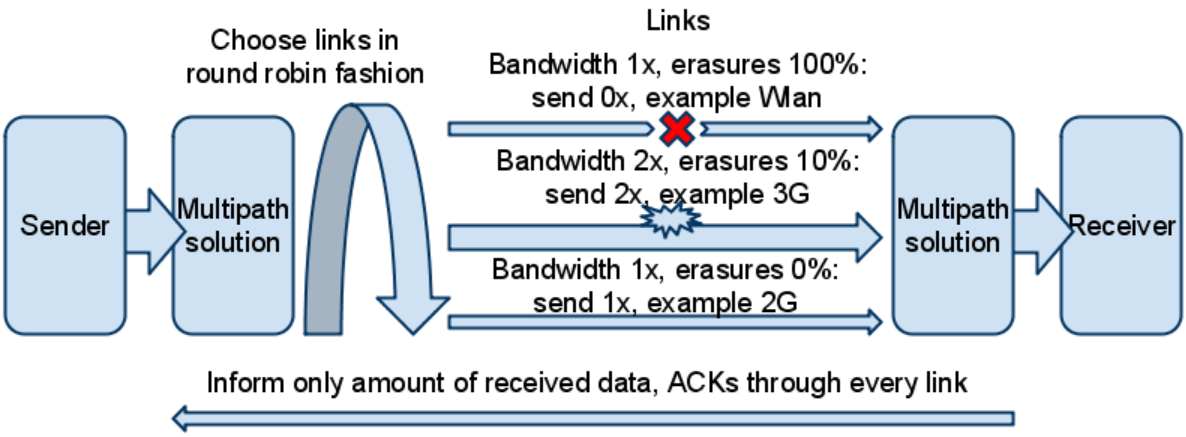

Fig. 1. General architecture of the FEC-multipath

The benefit of using FEC is that the protocol does not have to keep record of lost packets. Sender transmits as much data over the links as possible. The receiver will simply send ACKmessages to inform how much it has received so far. The sender can make estimation how much to send data based on these ACK-messages. FEC encoding enables decoding of the original data with any subset that is sufficiently large. By diving the original data to blocks it is possible to transfer streaming data as well as files. Using blocks causes small buffer at the both side of the transmission.

The packets are then encapsulated with the headers for Multipath support that this research devised and then sent through DCCP over UDP.

III. Protocol Stack

With this research, we enter into a multipath paradigm of the protocol stack as in figure 2. Unlike the traditional single path transport, the multipath transport shall apply headers to the incoming packets and send them via different links and thus multiple branches of DCCP, UDP and IP encapsulation will take place at a host. Different instances of encapsulation shall take place for a packets going via different links. For example, a packet going through WLAN link will have a different DCCP, GUT and lower layer protocol actions than a packet that shall be forwarded through ethernet. Both packets, however will be treated by the same multipath layer process.

The DCCP layer has been built to take advantage of it's congestion control capability. Since we are using UDP which has no congestion control mechanism, the network health is very effected, especially at the first hop which is our foucus area. Thus DCCP is appropriate here. Then after follows the regular protocol stack for each of the network interface of the end host.

A client server communication approach is shown in figure 3, which clearly shows that each link is preserved and the communication and signaling occurs via it.

The multipath layer lets the application layer view the connection in a single link perspective. It is also responsible in ensuring the reliability of the data transfer using it's signaling mechanism described in section 7. It inserts incoming packets into windows and then encodes them with FEC data.

The DCCP headers provide concestion control features and UDP transport will allow these packets to pass Firewall and NAT's. GUT[8] allows DCCP to be carried over UDP by means of tunneling. The packets coming from different network interfaces will bear different source

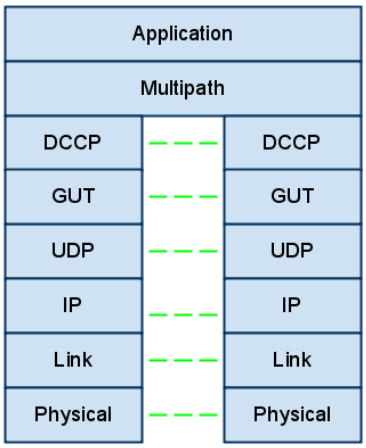

Fig. 2. A new realm of Protocol Stack

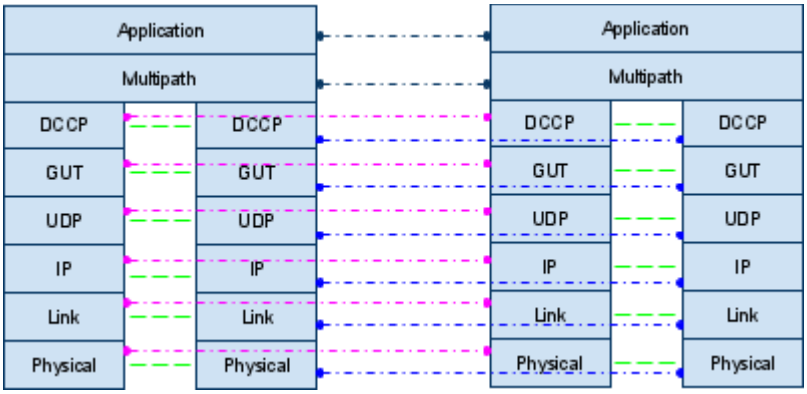

Fig. 3. A client server approach

addresses which will be aggregated at the destination and reordered. Thus, subsequent treatment by the IP layer, link layer and physical layer will follow on a per link basis.

VI. Forward Error Corrections

Forward error correction is method where the sender transmits extra redundant data which the receiver can use for replacing missing or correcting erroneous data [7]. Redundant data is encoded in such a way that it can restore any erasure anywhere in the original data.

Building the multipath with retransmission policy requires complex signaling system. This kind of protocol has to keep track which packets are lost. The latencies of the multiple links varies, thus this causes packet reordering which renders TCP-like reliability useless.

With FECs we are able to design a protocol where the end-hosts are not required to identify the lost packets. Every FEC-encoded received packet contains necessary data for receiver.

The only required information is the amount of data the receiver still needs for decoding the original content. This simplifies the signaling to simple ack-messages containing amount of received data so far. This also causes all received packets to increase the logical bandwidth, no duplicates are ever sent because retransmission are missing.

Considering a situation with three similar links. If one goes missing, one-third of all packets are lost. With TCP-like reliability this size of erasure rate is unbearable. With FEC-

encoded packets the erasure rate does not matter, only the amount of received packets is important. This can be achieved with simple acknowledgement messages.

We run our implementation with Nokia 770 which corresponds closely to the real mobile phone. With this device the encoding and decoding speed was over the possible network speed of the device.

The device was able to encode over 30 Mbit/s even with the systematic Reed-Solomoncoding. Test were done with different sizes of blocks, amount of original blocks and amount of transmitted blocks. Decoding results are omitted because the decoding is faster process.

There exists multiple FEC-algorithms. Most famous are the Reed-Solomon code [5] and fountain codes like Raptor [6] and LT [3]. Even though these algorithms are based on different ideas and implementations, the basic information that the sender and the receiver have to know is quite similar. A protocol which embodies FEC-algorithms has to be stateful, because some acknowledgments from the receiver are necessary [1].

V. DCCP OVER UDP

DCCP[2] is a mechanism to protect the health of the Internet and avoid congestion collapsein today's scenario of increasing UDP traffic. It provides the unreliable transport required by modern day real-time traffic and streaming media while providing congestion control mechanisms. It is fairly robust against primitive attacks. It is designed to be self sufficient but can be used alongside and over other transport protocols as well.

Using DCCP [2] for the congestion control simplifies the protocol remarkably. DCCP is simple solution for congestion control without reliability. Thus, we do not need to implement congestion control ourselves. There exists also Internet draft for encapsulating the DCCP inside an UDP [4], this would allow it's usage with current Internet's standard transport. Also there is a mechanism for adding different protocols over UDP by using a GUT(Generic UDP Tunneling)[8].

There are several congestion control schemes in DCCP and it is possible to set it at the beginning of the connection. This way we also divide the congestion control and reliable transport to two different layers and we can focus only on the multipath reliability and aggregation.

VI. MULTIPATH

After applying FEC into the incoming packets then arises the problem of selecting the link to forward the packet through among the multiple available ones. A signaling method has been devised for this protocol so as to ensure and support the reliable data taransfer.

The multiple links shall be initiated in a round robin fashion and both participating hosts monitor the link state and the erasure rate of each link. Then depending on the erasure rate, the sender will send different amount of packets via each link so as to let FEC counter the effect of the loss.

The protocol also implements a multiple window transmission scheme that allows continuous transfer of packets without having to wait for acknowledgements. We include a separate header structure for this purpose which will allow the proper functioning of the protocol and it's mechanisms. These specific details shall be published in the draft of the protocol.

The protocol is made modular in terms of the FEC block, windows, multipath selection mechanism and signaling will bind them all together.

VII. SIGNALING AND RELIABILITY

Our Data message contains: generic data header, packet seq number, link id, options(timestamp and current RTT estimation). Feedback message contains: generic ack header, and for each chunk describing a link:(timestamp, delay, reception rate and loss rate estimate)

The sender keeps sending packets and does not wait for acknowledgement. The receiver sends back acknowledgements at regular intervals through each of the links in a round robin fashion. If the sender is only waiting for the acknowledgement, the bandwidth is lost. It would make sense to continue sending new data even if the ack-message for last data block has not yet arrived. This is the reason for a multiple windows architecture. The sender will send data of the next window when it's predicting it has transmitted enough for the last one and is waiting for the ack-message. The signal flow follows as in Figure 4.

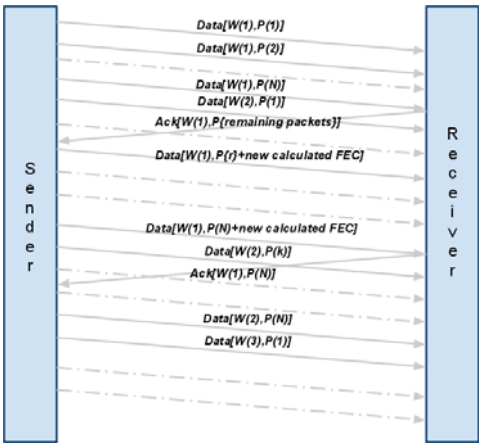

Fig. 4. Signaling Flow.

Signaling also includes the erasure rate estimation of the links by the receiver and sending of that information back to the sender. The sender then adapts accordingly.

At the receiver side, a packet loss is considered when 3 or more packets with higher sequence number than the lost packet arrive. The loss rate is calculated taking n amount of most recent packets arriving at the receiver.

The receiver keeps track of how many packets out of the last n packets were received(nrecv) and how many were lost(n-loss). The packet loss rate then could be easily calculated by n-loss/n.

During the startup of transmission through a link, the value of n should also start from zero and increase with the sequence number of received packet till at least n packets arrive.

At the senders side, the erasure rate information obtained in the feedback message is utilized to change the packets sent through that link. The amount of FEC encoded data is varied in proportion to corresponding erasure rates in the links so that we have enough redundant data to mitigate that error. The multipath layer will use the information of the erasure rates and the history as well to add or decrease the amount of FEC data.

For packet loss rate more than (say 50 percent), we can consider the link unusable and take it as a link failure. The sender then starts sending 0 packets through that link and use other remaining links.

It will however be monitoring that broken link and whenever an acknowledgement(with the stale erasure rate info) arrives via that link the sender assumes that the link is alive again (if down) or the erasure rate might have decreased(if it had crossed the threshold). The transmission through that link is then resumed and a new erasure rate calculation of that link takes place at the receiver and so on.

There are two ways of implementing reliability over multiple links [9]. Path-dependent and Path-independent. If the data is split into different paths and reliability is built on each path/flow, we would have path-dependent scheme. This is efficient if we want to correct single packet or small erasure in channels. However we are also considering lost links. All data sent to the missing link is lost and the path dependent reliability cannot recover that data.

Path-independent reliability does not take advantage or know that there exist multiple links. Reliablity system has no knowledge which link contains the erasure. This way the reliability can be seen same as in singlepath protocols. Thus, this protocol uses path-independent reliability.

The idea of reliable signaling is simple due to the usage of forward error correction. The receiver has to get enough data so that it can recover the desired packets back by decoding at the receiver. There is no need for TCP-like reliability where the order of the received packets and identifications of the missing one are required. Receiver will inform the Sender with Ack-message after it has got enough data packets. That message will contain information about the amount of data received from a particular window number by using offsets.

All data exceeding the required amount is unnecessary and redundant. Yet the sender cannot know the exact amount of data receiver has got. It only has the amount the receiver has lastly informed and statistics about the history. This causes some unnecessary transmitting. The sender can keep record of the history and count of the transmitted packets, thus it might stop transmitting after it has prediction that enough packets were sent.

This path indepent reliability along with the appropriate signaling wherever needed will form the basis for the FEC assisted multipath transport.

VIII. Implementation and Simulation

A python implementation has been done to observe the behaviour in a single link and it works as in Figure 5. Current implementation uses tun devices to catch packets from the network. Packet Windows -class gets as many packets as it wants from the sniffers buffer and make special windows we are using. Windows consists of normal IP-packets (and some redundant data that FEC adds, if needed.).

Proxy loops through all of our windows and when some data is found from some window it gets as many bytes from there as it wants. Proxy adds data to our own header and sends data to other side using UDP. Other side rebuilds our packet windows and after all packets are ready it separates ip-packets from the window and forwards packets.

Now we are moving the implementation in C. A method devised to send different protocols over UDP [8] and thus carrying our protocol as well as DCCP will be used. This shall be attempted and some kernel patches shall be applied wherever deemed necessary.

A simulation environment is also being constructed in NS3 where the performance of the protocol is being tested and compared with existing protocols like TCP, SCTP and the like. Better bandwidth utilization and increased reliability is anticipated with not much latency degradation.

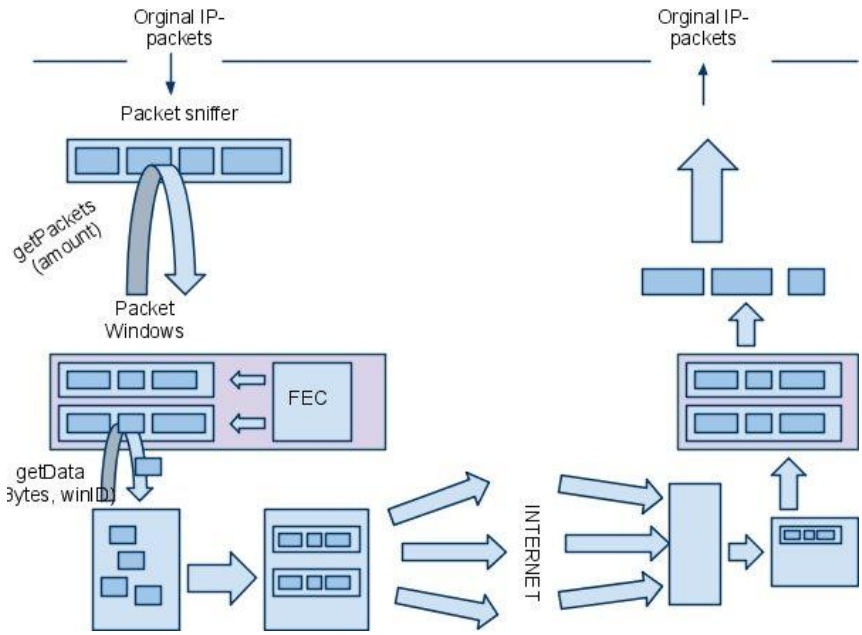

Fig. 5. Implementation Block Diagram.

IX. PROTOCOL USAGE

This solution could be used to improve mobile devices web experience. The client is the mobile device web browser and the server side is a private server with a web proxy. The mobile device can use the multipath protocol to connect this private server. With this scheme the mobile device can browse the web with utilizing multipath on the wireless transport.

Another use case is VPN; we can join two networks with VPN and use multipath protocol to tunnel the VPN traffic between two networks. The protocol can be applied at the gateways which will do the task of distributing the packets via multiple outgoing paths as well as aggregating the packets coming from multiple paths.

X. CONCLUSION

This research proposes a multipath reliable transport with forward error corrections on top of DCCP. It is possible to create multipath solutions based on proxy servers and tunneling and the transport protocols need not to be modified.

We are currently implementing a proof of a concept system where we will use multiple network connections with real time applications. Our research seems to indicate that with simple multipath solution it is easy to increase the whole network reliability and link aggregation. This is possible with using cheap already existing hardware and protocols and building only a new layer on top of the protocol stack. This perspective give us possibility to widen the current Internet and cellular usage without modifications to current infrastructure.

REFERENCES

[1] P. Ghosh, K. Basu, and S. K. Das. Improving end-to-end quality-of-service in online multi-player wireless gaming networks. Comput. Commun., 31(11):2685–2698, 2008.

[2] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340 (Proposed Standard) Mar. 2006.

[3] M. Luby. Lt codes. In FOCS ’02: Proceedings of the 43rd Symposium on Foundations of Computer Science, page 271, Washington, DC, USA, 2002. IEEE Computer Society.

[4] T. Phelan. Datagram Congestion Control Protocol (DCCP) Encapsulation for NAT Traversal (DCCP-NAT), Oct. 2008.

[5] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. In Journal of the Society for Industrial and Applied Mathematics, volume 8.2, pages 300–304, June 1960.

[6] A. Shokrollahi. Raptor codes. IEEE/ACM Trans. Netw., 14(SI):2551–2567, 2006.

[7] S. B. Wicker. Error control systems for digital communication and storage. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[8] J. Manner. Generic UDP Tunnelling (GUT) draft-manner-tsvwg-gut-00.txt, July 30, 2009.

[9] T. Ming-Fong., K. Chun-Yi.,K. Chun-Nan.,S. Ce-Kuen.

Multi-path Forward Error Correction Control Scheme with Path Interleaving http://hdl.handle.net/2377/10812, August 5 2008.