Multipath BitTorrent over SCION

Marten Gartner
Otto-von-Guericke University
Magdeburg, Germany
marten.gartner@ovgu.de

Thorben Krüger
Otto-von-Guericke University
Magdeburg, Germany
thorben.krueger@ovgu.de

Martin Koppehel
Otto-von-Guericke University
Magdeburg, Germany
koppehel@ovgu.de

David Hausheer
Otto-von-Guericke University
Magdeburg, Germany
hausheer@ovgu.de

***Abstract*—In this work, we extend an existing BitTorrent over SCION implementation to improve download performance through the use of multiple paths to each available peer. With our concept of path-level peers, the introduction of multipath communication with each peer requires no change to BitTorrent's mature file sharing algorithms.**

The path-level peers are derived from peer addresses and the set of possible paths to each of them is processed to select paths. For the experiments in our local high-performance testbed, we use a simple, hop-count-based heuristic to pick a specific number of suitable paths for each peer. For real-world setups, we implement an upload-based, disjoint path selection, that avoids using shared bottlenecks in multiple paths.

By keeping BitTorrent's built-in parallelism in our multipath implementation, we achieved a performance increase of factor two for aggregating two paths in our high-performance testbed, outperforming a path-unaware BitTorrent over IP approach. In the real-world testbed, we observe a 2-3 times higher throughput for our proposed path selection compared to a naive approach.

***Index Terms*—Multipath, BitTorrent, Path-aware networking, SCION, File transfer**

I. Introduction

The Internet was conceived decades ago, and some of the original design decisions have had unfortunate side-effects and security implications that persist to this day. So far, from the perspective of a mere host, no control over the path that is used to access a resource over the Internet is possible. There is not even any guarantee that a packet sent from a given source to a given destination will always use one particular path. Neither is there a general mechanism for splitting traffic across multiple different paths to a specific destination, which promises benefits for applications through avoiding congestion and aggregating bandwidth over multiple paths.

While resilience against DDoS attacks or route hijacking may be among some of SCION's major selling points, our work concerns itself with its promise of full and transparent path control for every end host, and easy multipath. From the perspective of a network host, such capabilities are a significant novelty. The possibility to freely select a particular path, or even multiple paths, to use for communication with a remote destination, creates interesting new opportunities for the host to handle traffic optimization and failure recovery by itself, autonomously.

Researchers have proposed different approaches to implement path-awareness. Some try to work within the limits of the current Internet architecture [12], [27], other attempts involve a complete redesign of the Internet architecture from scratch, to avoid repeating the old mistakes. While there are a number of path-aware approaches [23], [28] with their own merits, our work focuses on the relatively mature, well-known and documented open source SCION architecture [30]

In this work, we will concentrate on a practical demonstration of one of the possible improvements that path-control could facilitate: Bandwidth maximization through the simultaneous use of multiple paths to a destination. To fully utilise the potential of this improvement, applications need to fulfill specific requirements, i.e., parallel processing of data to benefit from path aggregation. Thus, we decide to combine SCION's path awareness and multipath features with the inherent parallelism in the design of the popular BitTorrent file sharing protocol. As an application protocol, BitTorrent has already attracted much attention from the research community, however mostly in setups containing a large number of peers and expansive network topologies [1], [2], often resulting in low performance. In this work, we choose a different setup, where fewer peers and high performance links are combined into simpler topologies in order to allow us to gain meaningful insights into the effects of host-based path selection.

The aim of this work is to demonstrate a minimally-invasive adaption of a BitTorrent client application to a realistic path-aware network environment, in order to illuminate some of the effects and possible improvements that a host-based path-selection scheme can have on bandwidth.

To this end, we provide the following contribution:

- With path-level peers, we propose a thin, minimally-invasive mapping of multipath capabilities to the BitTorrent protocol, that otherwise retains all key properties of BitTorrent's underlying, mature peer2peer algorithms.
- We develop a disjoint path selection algorithm, that avoids shared bottlenecks while increasing fairness in the distribution of network resources.
- We benchmark BitTorrent over SCION with multipath support against BitTorrent over regular IP and evaluate the performance of our disjoint path selection strategy compared to naive approaches.

The remainder of this work is structured as follows: In Section II we provide background for SCION and BitTorrent, followed by related work in Section III. We discuss the implementation of multipath support for BitTorrent over SCION in Section IV and present relevant benchmarks from our high-performance testbed in Section V. Afterwards, we present a disjoint path selection algorithm in Section VI followed by its

evaluation in Section VII. We conclude and provide outlook for future work in Section VIII.

II. Background

In the following, we provide a brief overview on the SCION architecture as well as on BitTorrent.

A. SCION

Nowadays, upcoming security issues need to be fixed in the current Internet, since its architecture was designed decades ago without proper knowledge about the majority of attack vectors. In order end up with an Internet with proper security guarantees and path control, we need to redesign the architecture from scratch. SCION is the most promising of these approaches. The SCION architecture [30] has been designed to allow retirement of the BGP protocol in the future Internet, addressing modern threat models at the fundamental protocol level and avoiding many current issues with single roots of trust, etc. SCION also comes with communication guarantees and path control capabilities, allowing applications to use one or more paths in parallel to a given destination, generally enabling multipath communication.

1) Network Structure: Traditional multipath approaches like MPTCP [20] and MPQUIC [4], define a path as an outgoing network interface. Consequently, they perform multipath by transferring data over multiple interfaces in parallel. SCION defines paths as a list of hops through the network to reach a particular destination. A SCION path contains one or more autonomous systems (AS). SCION networks are separated into different isolation domains, ISDs, which can correspond to, e.g., geographical regions, legislative domains (like a single country), company or research networks.

Each ISD contains at least one AS. One or multiple ASes are running as Core ASes and form the ISD Core, which builds the trust root configuration TRC. The TRC provides basic cryptographic information like secrets and keys, to which all other ASes must agree. Each AS has a private key and a certificate to authenticate itself to different SCION services on the control plane as well as the data plane. Connections in the SCION architecture can be either native SCION links or overlay SCION links, which are tunneled over normal IP links. Links between Core ASes are referred to as core links. Usually, ASes are connected in a way to link them to Core ASes, either directly or via intermediate ASes. Direct links between two non core ASes are referred to as peering links

Figure 1 shows an example SCION network topology. This example contains two ISDs ISD 1 and ISD 2. Each of them contains at least one Core AS, Core AS 1 and Core AS 2 for ISD 1 and Core AS 3 for ISD 2. All regular ASes are directly or indirectly connected to their core ASes. Furthermore, AS 6 and AS 7 are connected with a peering link crossing the borders of both ISDs. As can be seen from Figure 1, a regular AS can have connections to more than one Core AS.

2) Path Construction: By distinguishing between data and control plane, SCION enables a clear separation of responsibilities for the different components. The control plane is responsible for routing decisions. For applications, one of the most important features of the control plane is the path lookup, where an AS queries available paths to a specific SCION address. A path lookup operation is performed to retrieve path segments, specifically a triplet consisting of:

- Up path segment: Path segment from local AS to core AS
- Core path segment: Path segment from core AS to remote ISD core AS
- Down path segment: Path segment from remote core AS to remote AS

To create a valid SCION path, any element of the path triplet may be omitted, depending on the network topology, e.g. a core path segment may not be required if a peering link exists on the path to the remote AS.

The main responsibility of the data plane is packet forwarding. To send packets to a remote destination over SCION, a path to this destination must be determined and encoded in the packets. Compared to the current internet architecture, this significantly reduces the state that routers have to keep, because the information where to forward the packet is stored in the packet itself. A path is defined as a combination of up path, down path and core path segments and is encoded in a special efficient format to reduce bandwidth overhead.

Figure 2a visualizes the path construction in a sample SCION network consisting of 6 ASes and 2 core ASes. In this example, AS A wants to send packets to AS F. Consequently, A starts a path lookup and retrieves the path segments depicted in Figure 2b, where the segment starting with info field INF1 represents the up path segment, the one with INF2 the core path segment and INF3 the down path segment. Each path segment contains an ordered list of hop fields. Each hop field represents a description of physical or virtual links between two interfaces, the ingress and egress interface. Furthermore, the hop field contains the AS signature, the message authentication code MAC. Additional info fields are contained in each hop field. For up and down paths, the

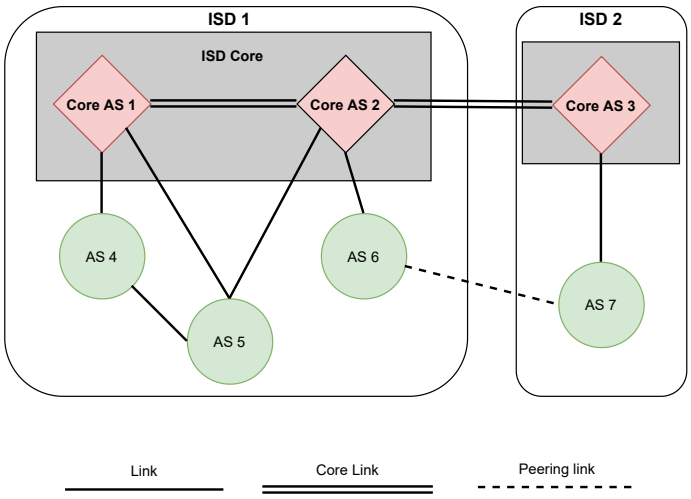

Fig. 1. Example SCION network topology consisting of two ISDs

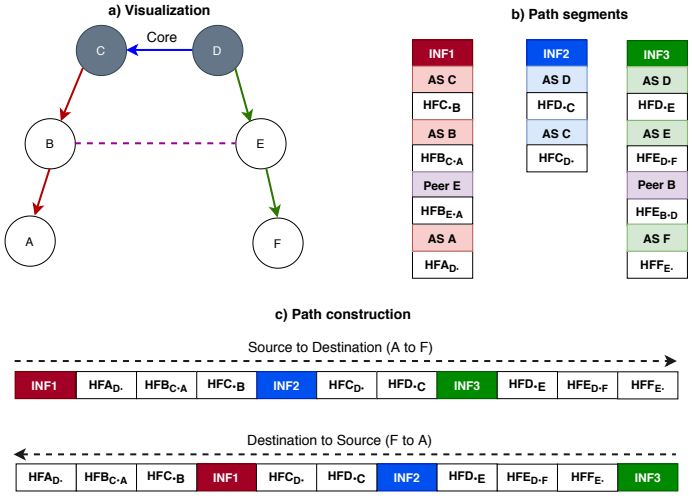

Fig. 2. SCION path encoding with a network visualization in a), the respective path segments response from AS A querying paths to AS F in b) and constructed paths from A to F and vice versa in c).

path segments start with the core AS and followed by the other ASes. Peering links are also encoded as hop field and returned from the path server, as can be seen in case of the link from AS B to E, With these path segments, one possible constructed path from A to F and reversed from F to A is shown in Figure 2c. (Note that since there is also a peering link, this is not the shortest path in terms of the number of hops.) The AS A now orders the info fields containing the hop fields between its location in the network and the location of AS F, thereby creating a valid path to F. The Figure finally also shows the path in a reversed view from F to A containing the same hop fields.

Now containing a valid path in each SCION packet, the packets have to be forwarded to their destination. On AS-level, this is usually done by the SCION border router, until they reach their destination hosts, entering the SCION host stack.

3) SCION Host Stack: The SCION control and data plane responsibilities are implemented through a set of services running on a single SCION end host as well as on the AS level. Each SCION end host runs at least the SCION Daemon, that is responsible for controlling the host, and the SCION Dispatcher, which is responsible for parsing all incoming and outgoing packets. The AS operates additional services such the local path server and the SCION Border Router.

All sent and received packets of SCION apps that run on endhosts are passed to a process called the SCION Dispatcher. AS-incoming SCION packets are handled by the border router and passed to the SCION Dispatcher, which forwards it to the target application. Outgoing SCION packets are then sent over the dispatcher to the SCION border router. However, the Dispatcher comes with performance limitations, which can be overcome by a proposed approach called xiondp [9].

B. BitTorrent

Through the P2P approach, BitTorrent aims to overcome the limits of classic client-server setups, where a central server can be a single point of failure.

The BitTorrent protocol specifies file transfer as a distributed mechanism between peers without the need for any central coordination. (Some initial way to exchange network addresses among peers is nevertheless required, e.g., by means of a tracker or, alternatively, a distributed hash table (DHT)). In BitTorrent, files typically are not transferred in the usual form of a single, continuous byte stream that contains the complete file. Instead, large files are split into equal-sized pieces (typically with a size between 32KB and 256KB). It is a key feature of BitTorrent that exchange of these pieces can be easily parallelized. BitTorrent peers that already have a complete local copy of a file are referred to as seeders, as opposed to leechers, which still have to obtain some or all of the constituent pieces of the file from the other peers. For each new file uploaded to the system, there must be at least one seeder, that initiates the distribution of the file. After each piece of the new file is uploaded to at least one other peer, the seeder is not further mandatory to distribute the file and new leechers can retrieve pieces of the file from other peers.

III. Related Work

The mature peer2peer mechanisms behind BitTorrent have made the protocol an attractive target for networking research in the past. Ren et al. present TopBt [22], an adaption of BitTorrent that uses proximities in addition to transmission rates to detect peers to collaborate with. Li et al. present UTAPS [16], a peer selection algorithm for BitTorrent that benefits from the awareness of underlying topologies. Similar to our approach, UTAPS uses the underlying topology to determine the hop count as metric to select peers. However, we perform this decision on a per-path level, allowing us to select multiple paths to each peer. Catro et al. propose BestPeer [3], a peer selection algorithm that supports multipath running on multi-radio, multi-channel wireless mesh network and show a reduction of download time of 40% compared to traditional BitTorrent. While this work enables multipath through different wireless channels, our approach is built upon a networking architecture that allows for multipath communication at Internet scales.

A lot of research was done with focus on IP multicast as an efficient way to distribute content to multiple peers without duplicating the traffic [5]. Since IP multicast requires a lot of state in switches and routers, the global distribution suffers [6], [21]. As an alternative to IP multicast, overlay approaches are considered: Bullet by Kostić et al [14] is an overlay approach to efficiently distribute files from a single source to a huge number of receivers introduced. Similar to BitTorrent, files are split into pieces and distributed to other peers.

Next to SCION, several other approaches to enable path control on the host exist. PathLet Routing by Godfrey et al [11] is an approach based on segmentation of inter-domain routes into path fragments. PathLet stores an encoded forward

rule in the next hop entry, which will be processed by the next router. Establishing multipath data transfer can also be realized completely on application level. Yu et al. proposed mPath [27], an algorithm and implementation to leverage proxies to create multiple paths to particular end hosts. Mpath uses similar approaches as SCION e.g. path lookup.

Multipath variants for transport protocols like TCP and QUIC exist in the form of MPTCP [20] and MPQUIC [4]. However, these protocols require the presence of multiple network interfaces on the host, over which the traffic can then be distributed. In contrast to SCION, no influence or guarantee of the actual traffic flow through the network is possible.

To detect shared bottlenecks, different approaches focusing based on MPTCP have been proposed, some via active measurements [7], [26], other via passive shared bottleneck detection [13]. These approaches are constrained by an inherent lack of information about the actual path that the data uses through the Internet. With Espresso [29], Google presented a BGP-based approach for traffic distribution and bottleneck avoidance at the edge of their network, rather than on the endhosts.

Further work shows that SCION is capable to achieve high performance, i.e. Neukom et al. propose Hercules [19], a protocol for very high performance bulk data transfer over SCION using AF_XDP and de Ruiter et al. present a SCION Border Router implementation in P4 [24]. Gessner et al [10] leverage SCION's multipath capabilities for a video streaming approach that distributes audio and video channels over different paths, collecting performance metrics to find the best suitable paths for each channel. Gartner et al. present an XDP-based approach to bypass the SCION Dispatcher including optimised SCION application libraries [9], which we use in our Multipath BitTorrent over SCION implementation.

IV. Design and Implementation of BitTorrent over SCION

As presented in the previous section, SCION provides path control for inter-AS traffic while guaranteeing, that the traffic flows along the chosen paths. However, to fully benefit from SCION's path-awareness, also applications need to support e.g. parallel data processing to increase performance of multipath communication. In this work, we choose BitTorrent as already parallelised application to unlock further potential by aggregating multiple SCION paths.

Normally, a BitTorrent peer is identified solely by its network address and port. This address could be either an IPv4, IPv6 or, in our case, a SCION address. We will refer to peers that are only described by their address as address-level peers for the rest of this work. To distinguish between multiple SCION paths to a particular peer, we introduce a new representation that we will refer to as a path-level peer. They are represented by the tuple $(addr, path)$, consisting of the peer's SCION address (including the port), together with one possible path to this address. Since IPv4 and IPv6 lack the necessary path-aware properties, path-level peers are always SCION peers.

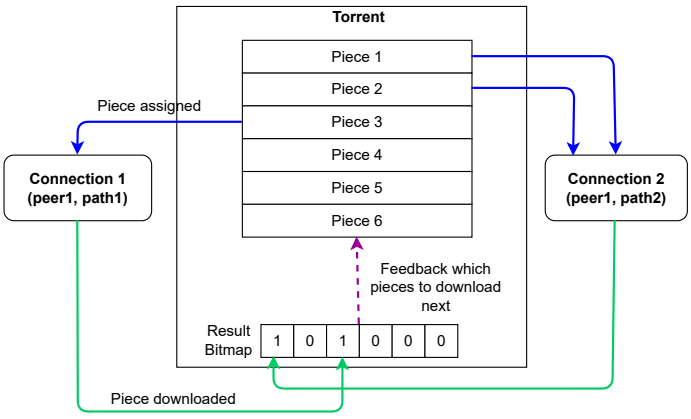

Fig. 3. Multipath implementation by downloading a torrent file over multiple paths (connections).

After introducing the concept of path-level peers, we present our path-aware BitTorrent implementation called BitTorrent over SCION, which treats different paths to the same peer as several, distinct, path-level peers. Instead of TCP as transport protocol, we use QUIC for BitTorrent over SCION. In all other respects, we keep the BitTorrent standard in our implementation.

Generally, peers that are returned from a tracker or added via static bootstrapping to BitTorrent are address-level peers, since they may not be SCION peers and do not have path information associated with them. To generate path-level peers from a given SCION address, an additional path lookup is required. If multiple paths to the peer are available, the corresponding number of path-level peers can be generated.

BitTorrent over SCION can be configured with an upper bound for the number of different paths to a destination. Without further configuration, the path selection is performed by determining the shortest paths by the number of hops with respect to the upper bound.

After mapping address-level peers to path-level peers, BitTorrent is capable of running its filesharing algorithm over each path independently. For each path-level peer, a QUIC [15] connection is established to ensure reliable transfer of data. We choose QUIC as modern protocol running on UDP, because TCP is currently not yet implemented for SCION. Figure 3 shows the download process pieces of a particular file. Each successfully established connection fetches piece information from a queue and requests the particular piece by sending piece request messages and wait until peers send back the requested pieces. After each received piece, the integrity of the retrieved piece is verified. For this, the piece is hashed to verify the resulting hash against the one stored in the torrent file for that piece. Retrieved pieces are stored in main memory until the file is downloaded completely.

Since requests and retrievals of pieces over different connections are handled in their own dedicated threads, pieces

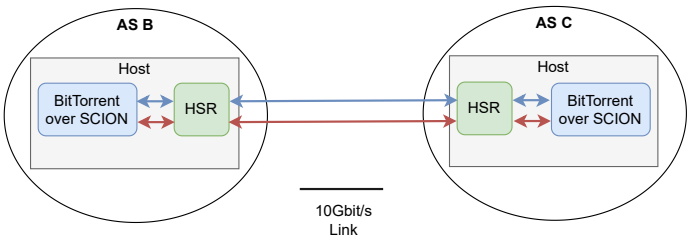

Fig. 4. Benchmark setup of two servers that run two SCION ASes connected via 10Gbit/s link.

are downloaded concurrently, which speeds up the process and improves the overall download bandwidth. The main thread is responsible to iteratively check the result bitmap for missing pieces. Once all pieces are retrieved, the main thread closes all connections to all connected peers that serve pieces of the current torrent and writes the file from main memory to disk.

V. Evaluation: Path Aggregation

After the above introduction of path-level peers and their implementation, intended to leverage the inherent parallelism of the BitTorrent architecture, we now verify that useful bandwidth can indeed scale in practice as expected. To this end, we compare BitTorrent over SCION using one and two paths against a path-unaware BitTorrent over IP implementation, both using QUIC as transport protocol.

To provide a reliable test setup for multipath BitTorrent over SCION in this experiment, a setup of multiple ASes connected via high performance links is used. Figure 4 shows the chosen network topology for the path aggregation benchmark. The selected topology consists of two servers connected via two direct 10 Gbit/s links connecting both servers. Each server has a 10 core Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz and 48 gigabytes of main memory. The used network cards are Intel X722 NICs. Each server runs Ubuntu 20.04 with a Linux kernel 5.4.0-62. This setup is used to evaluate the performance of multipath BitTorrent over SCION for several reasons. At first the servers provide high performance CPU and network cards to allow multi-gigabit bandwidths. Secondly, with 10Gbit/s links, possible limitations of BitTorrent regarding the maximum available bandwidth on single links can be evaluated. This topology is a good candidate to represent our targeted use case scenario of having a low number of fast connected peers. Both ASes run proprietary high-speed SCION routers (HSR) [25], to avoid certain known bottlenecks in the open-source router implementation. The high performance experiments use xiondp [9], an XPD-based approach to bypass the SCION Dispatcher which is a known bottleneck for higher throughputs.

In this experiment, the bandwidths achieved while downloading a 5 Gigabyte file are measured for singlepath and multipath BitTorrent over SCION and a baseline implementation over IP. The benchmark is executed using jumbo frames with a MTU of 3000 (maximum MTU supported by XDP, due to the usage of xiondp [9]). The singlepath benchmark candidates use

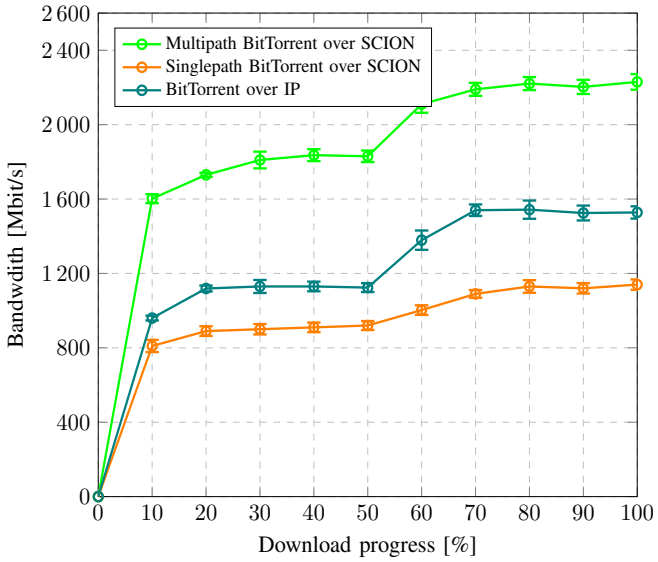

Fig. 5. Comparison of the average download bandwidth of all benchmark candidates downloading a 5 Gigabyte file with 5 repetitions expanded by the resulting standard deviation.

one direct 10 Gbit/s link, whereas the multipath candidate use both links in parallel. We expect BitTorrent over IP to perform better than Singlepath BitTorrent over SCION, due to the additional overhead of SCION. Since QUIC is not compatible with the optimised library of xiondp (as presented in [9]), the overhead results from the upstream SCION library for packet processing. However, by aggregating the bandwidth of 2 paths compared to using only one path, a twofold increase in performance is expected for Multipath BitTorrent over SCION. Consequently, we expect Multipath BitTorrent over SCION to outperform BitTorrent over IP significantly.

Since the download time differs significantly, the bandwidth of all candidates is measured at intervals during the progress of the download. This allows a comparable presentation of all candidates. Figure 5 shows the comparison of the average download bandwidth of all benchmark candidates. Again, the measurements are repeated 5 times. The standard deviation is presented as error bars around the data points.

As expected, Multipath BitTorrent reaches the highest bandwidth, with around 2200 Mbit/s. The singlepath SCION candidate reaches around 1100 Mbit/s, significantly slower than BitTorrent over IP with 1500Mbit/s. Although 10 Gbit/s bandwidth is available per path, BitTorrent over SCION achieves only a fraction of this. We suspect potential bandwidth limitations in the used QUIC implementation [17], which are used under the hood of all candidates, resulting in not fully utilizing the available bandwidth. Furthermore, we observe an increase of bandwidth at 50% download progress for all candidates, which we assume to be reasoned by QUIC. With this experiment, we verify that with adding path-level peers, a performance increase of factor two for using two paths instead of one can be achieved. Furthermore, we demonstrate that using multipath based on SCION, we were able to outperform

BitTorrent over IP.

In this small, high-performance testbed, our expectations were confirmed. However, moving this adaption to larger testbeds imposes another important problem: Selecting the shortest paths (determined by the number of hops) could lead to a choice of paths that may (partially) overlap and thereby potentially impinge on each other's useful bandwidth by compounding congestion at shared bottlenecks.

VI. Upload-based Disjoint Path Selection

As presented in Section IV, per default Multipath BitTorrent over SCION uses the hop count as metric to select a number of paths with a configurable maximum amount. When this limit is set suitably high, all available paths to each peer are considered. However, the overhead of using a large number of paths may eventually start to negate any benefits of aggregating their bandwidth. Furthermore, multiple paths may share the same bottleneck, making it pointless to aggregate them in hopes of increasing overall performance. To address this, we use built-in SCION capabilities to implement an improved, disjoint path selection strategy, aiming to avoid such shared bottlenecks.

In BitTorrent over SCION, our improved path selection strategy relies on two core assumptions: 1) Different paths that share the same hop may also share a bottleneck at this hop and 2) Peers that offer pieces to others are in a good position for path selection decisions with knowledge about all downloading peers, allowing them to strategically distribute their outgoing traffic via disjoint paths.

We implement a disjoint path selection strategy by searching for overlaps in all paths and discard all but one of the paths that share the same hops in deference to assumption 1). With respect to assumption 2), we decide to delegate path selection exclusively to the uploading peer. Both in combination promise to outperform naive path selection approaches. The connection setup and the path selection are implemented on top of SCION Pathdiscovery [8], a path-aware library for SCION.

As presented in Section II, a SCION path consists of multiple hops. Each hop contains ingress and egress interfaces of the respective AS. In SCION, the interfaces are represented numeric IDs, that are unique within the given AS. The combination of the AS identifier with the interface number results in a globally unique interface ID. In Figure 6, we show an intuitive mapping of a visual representation of hops and their interfaces to a list of such interface IDs.

The disjoint path selection happens in two steps: First, obtain a list of possible paths to a destination. For a given path, discard any other path that shares any of the original path's interface IDs. Next, from the paths that are still left, pick the next path and repeat the process until there are no more paths left. In case the local AS has only one outgoing interface, its ID is excluded from the selection process, to avoid reducing the number of paths to one while having suitable paths available. The same applies to the incoming interface ID of the destination AS.

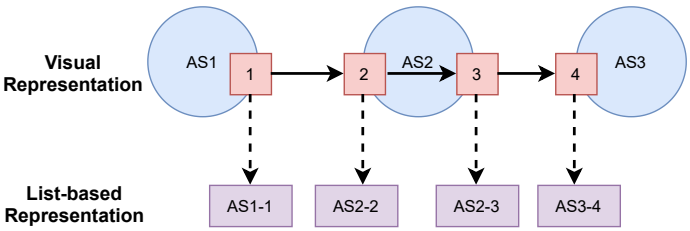

Fig. 6. List-based representation of hop interfaces using unique ids to perform conflict detection.

The uploading peer keeps a global list of currently connected peers and their assigned paths, helping to select a path set for a newly connected peer. To achieve this, an interested peer connects over the first available path to the uploading peer and waits for it to connect back. The uploading peer now applies it's disjoint path selection and connects back to the interested peer over the selected path set. The list is searched for overlapping paths, (identified by occurrence of the same Interface ID,) and the application re-distributes all possible non-overlapping paths in a fair manner to all connected peers. For example, if the store contains one peer PeerA that uses three paths and a new peer PeerB connects that has two available paths, with one of them overlapping with one of PeerA's paths, the path-selection process results in PeerB getting connected on both paths, with the overlapping path being dropped for PeerA to avoid a potential shared bottleneck, resulting in both peers using two paths each. The path selection aims to achieve a similar number of paths for all peers. However, newly connected peers have a significantly higher number of available paths than the connected ones. If conflicts with paths of connected peers occur in this case, the disjoint path selection does not remove paths from the connected peers but uses alternative paths to the new peer.

VII. Evaluation: Path Selection

We now evaluate the effects of our disjoint path selection strategy on download performance in a testbed using one and two leechers.

A. Disjoint path selection to a single leecher

We validate the disjoint path selection strategy against the naive approach in a scenario with up to three possible paths between a single seeder and a single leecher. The naive approach uses the two shortest paths, determined by the number of hops.

Figure 7a) shows the experimental setup to test BitTorrent over SCION's disjoint path selection to a single leecher. The setup consists of 7 ASes. The two ASes B and C provide uplinks to the ASes A and D. Furthermore, the ASes E and G are connected via F and provide uplinks to A and D. We create a shared bottleneck on the interface connecting A and B, limiting it's provided bandwidth to 10Mbit/s. A host in AS A will receive a file via BitTorrent over SCION from a host in D. In this setup, SCION provides 3 paths from D to A. BitTorrent over SCION only uses 2 of the available 3 paths.

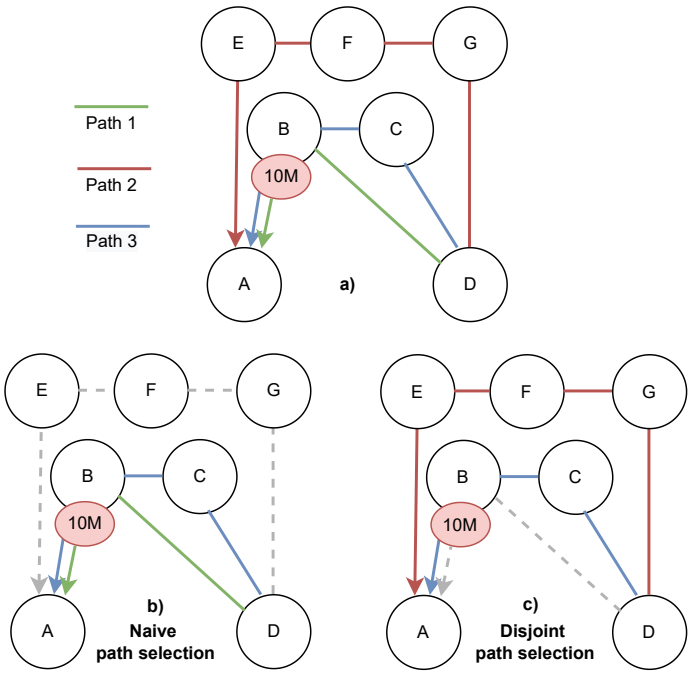

Fig. 7. a) Experimental setup for path selection with a single leechers, b) choice based on naive path selection, c) choice based on disjoint paths

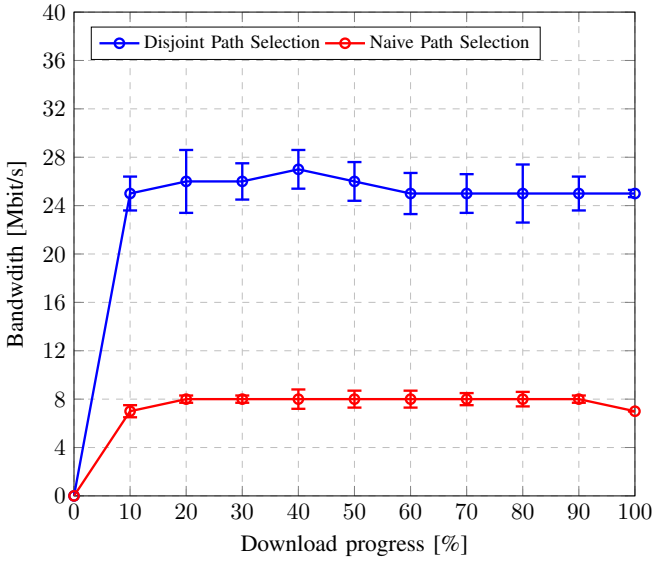

Fig. 8. Goodput of BitTorrent over SCION using naive and disjoint path selection to a single peer

All ASes are deployed in virtual machines equipped with 2 cpu cores.

Figure 7b) shows the selected path set of a naive path selection, using both paths that share a bottleneck. Figure 7c) shows the result of BitTorrent over SCION's disjoint path selection, avoiding the overuse of the bottleneck. We expect BitTorrent over SCION's path selection to deliver a twofold increase in overall bandwidth compared to the outcome of the naive path selection.

The experimental results for downloading a 100 MB file over five repetitions are shown in Figure 8, depicting the useful download bandwidth (i.e., goodput) of BitTorrent over SCION to a single peer using the naive versus the disjoint path selection strategy, with each approach aggregating exactly two paths. For easier comparison, the measurements are normalized to download completion percentage on the x-axis. Since the measured goodput by definition excludes all protocol overhead (most significantly that of QUIC), we can not expect to achieve the full 10MBit/s at the shared bottleneck. Indeed, the measured bandwidth is closer to 8Mbit/s. Also as expected, the naive path selection congests the shared bottleneck and thereby only achieves those 8Mbit/s of goodput in total. The disjoint path selection avoids creation of a shared bottleneck, reaching around 26Mbit/s, 8Mbit/s for the path over AS B and 18Mbit/s for the long path over ASes E,F,G.

As expected, the disjoint path selection is able to outperform a naive approach by avoiding using two paths with the same shared bottleneck. In our experimental setup, the disjoint path selection improves the performance by factor three compared to the naive one. However, in this example, also the leecher could perform a similar path selection decision resulting in the same output. Therefor, we demonstrate the benefits of performing path selection on the uploading peer in an experiment with multiple leechers.

B. Disjoint Path Selection for multiple Leechers

This experiment aims to show the benefits of moving the responsibility of path selection to uploading peers, in this case seeders. Initially in SCION, the applications that dial (client) to particular listening applications (server) are selecting which paths to use. In our case, this may lead to multiple downloading peers select paths that share bottlenecks. The uploading peer, however, has information about all connected clients and can detect shared bottlenecks in all available paths to all downloading peers.

Figure 9 shows our experimental setup to compare upload-based path selection against path selection performed by leechers. This setup consists of 5 ASes. The 3 ASes A, E, and D are connected to the two ASes B and C providing upstream bandwidth. BitTorrent over SCION hosts in both ASes A and D receive a file from the seeder running in AS E. For each AS A and D, 2 different paths are available to E. The ASes B and C are running on the same machines as used for our high-performance experiment. The ASes A, D and E are deployed on virtual machines in the hetzner cloud with 2 CPU cores and 4 Gbyte RAM and connected via SCION overlay links to B and C. Both leechers (A, D) use 1 path to receive the file from E. The upstream bandwidth of both ASes B and C is limited to 10Mbit/s, respectively

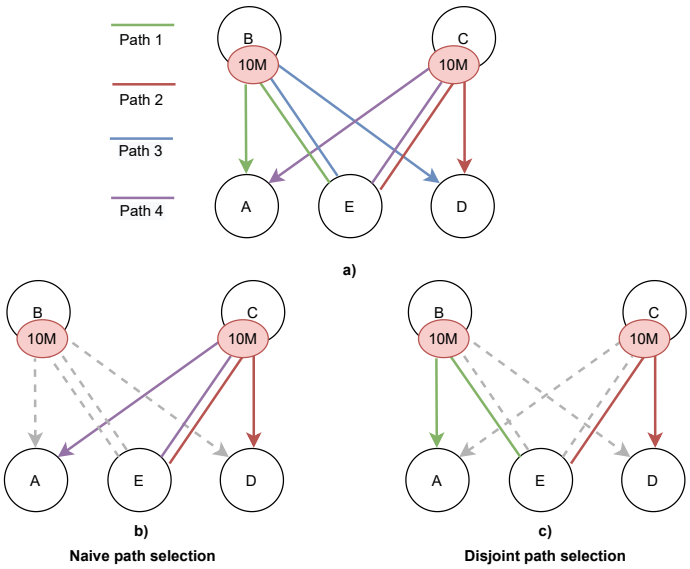

Fig. 9. a) Experimental setup for path selection with multiple leechers, b) naive, leecher-based path selection, c) upload-based, disjoint path selection.

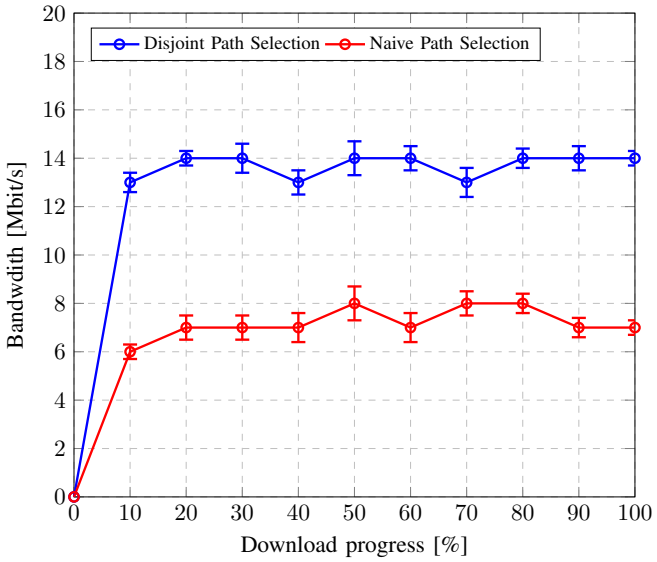

Fig. 10. Goodput of BitTorrent over SCION using leecher-based and upload-based, disjoint path selection to 2 different peers.

We expect the upload-based, disjoint path selection to perform better than the leecher-based path selection, which we expect to be limited to 10Mbit/s throughput, which means for BitTorrent over SCION around 7-8Mbit/s goodput.

In Figure 10, we present the results of comparing leecher-based, naive path selection to upload-based, disjoint path selection using two leechers and one seeder. We use a 100 Mbyte file for this experiment and repeat the transfer 5 times. Again, due to differing time required to download the files in both setups, we use the download progress in percent for the x-axis. As expected, if the leecher performs the path selection in a naive way, both leechers use paths that share the same bottleneck. This is observed by the naive approach only reaching 7-8Mbit/s goodput, whereas the disjoint path selection done by the seeder reaches around 14Mbit/s.

As expected, moving the path selection to the uploading peer, in this case a seeder, in combination with avoidance of shared bottlenecks outperforms a naive, leecher-based approach by factor two in this setup. The seeder knows all paths to both peers and is able to find shared bottlenecks in them, even if the paths are not targeted to the same SCION AS. In this example, the overall performance is doubled by avoiding to use paths over the shared bottleneck to both peers.

VIII. Conclusions and Future Work

In this work, we adapt BitTorrent to benefit from host-based path-awareness features like those offered by the SCION network architecture. We developed the notion of path-level peers, which allowed us to enhance an existing BitTorrent implementation with SCION support to add multipath features, with minimal modifications to the underlying file-sharing algorithms. We evaluated Multipath BitTorrent over SCION in combination with various additional optimizations against a singlepath SCION and a singlepath IP baseline. We demonstrated that, as expected, aggregating two SCION paths results in a twofold increase of performance compared to using one path, easily outperforming BitTorrent over IP despite some additional SCION-related overhead in our local testbed. In our path selection experiments, BitTorrent over SCION's path selection outperforms a naive approach by factor 3 in a setup with one and leecher and by factor 2 with two leechers. We observe that disjoint path selection done by the uploading peers, in our case a seeder, performs better with multiple leechers than each leecher performing an isolated path selection. Overall, we were able to show that SCION's path-awareness features can bring significant benefits to certain networked applications which can benefit from parallelization.

A further improvement for BitTorrent over SCION could be to extend the tracker implementation to pre-select particular path-level peers. In case the tracker is filled with the gathered data from the peers, this approach may benefit of shared knowledge about paths. To fill the tracker with information that are beneficial for path selection, e.g. which peers are connected over which paths, peers need to actively communicate this information to the tracker.

To overcome the per-connection limit, replacing QUIC as transport protocol is a promising option. Due to being optimized for downloading multiple small resources over one connection, QUIC does not fit the use-case of high-performance file transfer well. An alternative would be PARTS [18], which is currently in prototype status, but is in active development to fit exactly this use-case, as shown in [9].

After removing potential performance bottlenecks in the QUIC implementation, we plan to analyze how BitTorrent's built-in fairness performs in path-aware networks. By providing peers the possibility to actively select one or more paths, new challenges arise to achieve overall fairness, e.g. to evaluate choking on path level, compared to choke peers.

Finally, our proposed path selection may be extended to allow multiple peers reuse the same paths, even if there are unused alternatives. To realize this, seeders can observe variation in bandwidth to all connected peers when adding new paths containing shared hops. If no decrease for any peer will be observed, no shared bottleneck is detected.

References

[1] Bharambe, A., Herley, C., Padmanabhan, V.N.: Understanding and deconstructing bittorrent performance. In: Proc. ACM Sigmetrics (2005)

[2] Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and improving a bittorrent networks performance mechanisms. In: Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications. pp. 1–12. IEEE (2006)

[3] Castro, M., Kassler, A., Avallone, S.: Bestpeer-a load-aware multi-path peer selection for wireless mesh networks. In: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). pp. 1–6. IEEE (2012)

[4] De Coninck, Q., Bonaventure, O.: Multipath quic: Design and evaluation. In: Proceedings of the 13th international conference on emerging networking experiments and technologies. pp. 160–166 (2017)

[5] Deering, S.E.: Rfc1112: Host extensions for ip multicasting (1989)

[6] Diot, C., Levine, B.N., Lyles, B., Kassem, H., Balensiefen, D.: Deployment issues for the ip multicast service and architecture. IEEE network 14(1), 78–88 (2000)

[7] Ferlin, S., Alay, Ö., Dreibholz, T., Hayes, D.A., Welzl, M.: Revisiting congestion control for multipath tcp with shared bottleneck detection. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. pp. 1–9. IEEE (2016)

[8] Gartner, M., Krueger, T., Koppehel, M., Hausheer, D.: Scion path discovery, https://github.com/netsys-lab/scion-path-discovery, 2022-01-11

[9] Gartner, M., Wagner, J., Koppehel, M., Hausheer, D.: XDP-Accelerated Packet Processing on SCION Endhosts. In: NOMS 2022 IEEE/IFIP Network Operations and Management Symposium. IEEE (2022)

[10] Gessner, J.: Leveraging Application Layer Path-Awareness with SCION. Master's thesis, ETH Zurich (2021)

[11] Godfrey, P.B., Ganichev, I., Shenker, S., Stoica, I.: Pathlet routing. ACM SIGCOMM Computer Communication Review 39(4), 111–122 (2009)

[12] Gvozdiev, N., Vissicchio, S., Karp, B., Handley, M.: On low-latency-capable topologies, and their impact on the design of intra-domain routing. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. pp. 88–102 (2018)

[13] Hayes, D.A., Ferlin, S., Welzl, M.: Practical passive shared bottleneck detection using shape summary statistics. In: 39th Annual IEEE Conference on Local Computer Networks. pp. 150–158. IEEE (2014)

[14] Kostić, D., Rodriguez, A., Albrecht, J., Vahdat, A.: Bullet: High bandwidth data dissemination using an overlay mesh. In: Proceedings of the nineteenth ACM symposium on Operating systems principles. pp. 282–297 (2003)

[15] Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J., et al.: The quic transport protocol: Design and internet-scale deployment. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication. pp. 183–196 (2017)

[16] Li, W., Chen, S., Yu, T.: Utaps: An underlying topology-aware peer selection algorithm in bittorrent. In: 22nd International Conference on Advanced Information Networking and Applications (aina 2008). pp. 539–545. IEEE (2008)

[17] lucasclemente: A quic implementation in pure go, https://github.com/lucas-clemente/quic-go, accessed 2020-08-18

[18] Marten Gartner: Path-aware transport over scion, https://github.com/netsys-lab/parts, accessed 2021-09-27

[19] Neukom, C.: High-Performance File Transfer in SCION. Master's thesis, ETH Zurich (2020)

[20] Peng, Q., Walid, A., Hwang, J., Low, S.H.: Multipath tcp: Analysis, design, and implementation. IEEE/ACM Transactions on networking 24(1), 596–609 (2014)

[21] Ratnasamy, S., Ermolinskiy, A., Shenker, S.: Revisiting ip multicast. In: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications. pp. 15–26 (2006)

[22] Ren, S., Tan, E., Luo, T., Chen, S., Guo, L., Zhang, X.: Topbt: A topology-aware and infrastructure-independent bittorrent client. In: 2010 Proceedings IEEE INFOCOM. pp. 1–9. IEEE (2010)

[23] Rothenberger, B., Roos, D., Legner, M., Perrig, A.: Piskes: Pragmatic internet-scale key-establishment system. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS'20) (2020)

[24] de Ruiter, J., Schutijser, C.: Next-generation internet at terabit speed: Scion in p4. In: Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies. pp. 119–125 (2021)

[25] Systems, A.: Anapaya systems, https://www.anapaya.net/, accessed 2020-08-22

[26] Wei, W., Wang, Y., Xue, K., Wei, D.S., Han, J., Hong, P.: Shared bottleneck detection based on congestion interval variance measurement. IEEE Communications Letters 22(12), 2467–2470 (2018)

[27] Xu, Y., Leong, B., Seah, D., Razeen, A.: mpath: High-bandwidth data transfers with massively multipath source routing. IEEE Transactions on Parallel and Distributed Systems 24(10), 2046–2059 (2012)

[28] Yang, X.: Nira: A new internet routing architecture. ACM SIGCOMM Computer Communication Review 33(4), 301–312 (2003)

[29] Yap, K.K., Motiwala, M., Rahe, J., Padgett, S., Holliman, M., Baldus, G., Hines, M., Kim, T., Narayanan, A., Jain, A., et al.: Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication. pp. 432–445 (2017)

[30] Zhang, X., Hsiao, H.C., Hasker, G., Chan, H., Perrig, A., Andersen, D.G.: Scion: Scalability, control, and isolation on next-generation networks. In: 2011 IEEE Symposium on Security and Privacy. pp. 212–227. IEEE (2011)