A Bottom-Up Investigation of the Transport-Layer Ossification

Korian Edeline, Benoit Donnet
University of Liège, Montefiore Institute, Belgium

***Abstract*—Recent years have seen the rise of middleboxes, such as NATs, firewalls, or TCP accelerators. Those middleboxes play an important role in today's Internet, and are now extensively deployed in various networks including corporate networks, Tier-1 ASes, cellular networks, and WiFi hot-spots.**

Unfortunately, despite the added value that they bring to networks, they radically change the transport paradigm from the legacy end-to-end principle, and drive increasing complexity *in the path*. The consequences of these changes are a wide variety of simple to subtle impairments to protocols and features, that in turn lead to the *ossification* of the network infrastructure. While the latter is now a well-known problem, its causes are not that much understood.

To fill this gap, we provide a more detailed explanation of the factors of the transport-level ossification, and we give insights on their prevalence in the wild. We extract path conditions by processing a large collection of observations of middlebox in-path packet manipulations, and we categorize the observed transport impairments based on the complications that they engender. We show that more than one third of network paths are crossing at least one middlebox, and a substantial percentage are affected by feature or protocol-breaking policies. Finally, we show that the majority of the devices that implements them are located in edge networks.

I. Introduction

It is now acknowledged that the initial end-to-end paradigm of the TCP/IP architecture, where both participants in a communication would assume that all exchanged information addressed to the other participant would remain untouched in-transit, has come to an end. This evolution was caused by the progressive introduction of *middleboxes*, i.e., network appliances manipulating traffic for purposes other than packet forwarding [1], going from "simple" NATs to complex multi-policy traffic engineering systems that alter packets up to the application layer.

Today, middleboxes are deployed in all possible networks, and their number continues to grow. Corporate networks account for as many middleboxes as traditional network equipment [2]. Tier-1 ASes are deploying more and more middleboxes [3]. Cellular networks also rely on strategically positioned middleboxes (e.g., Carrier-Grade NATs) [4]. Most Customer-Premise Equipment (e.g., Home Gateways) also implements middlebox policies [5], [6]. Moreover, the introduction of Network Function Virtualization (NFV) and the recent progresses of virtualization technologies (i.e., hardware virtualization, containerization) have greatly facilitated and popularized middlebox development and deployment [7].

Regrettably, middleboxes have also been shown to engender multitude of connectivity, performance, and security issues. Establishing TCP connections with Explicit Congestion Notification (ECN) enabled can lead to connectivity blackouts [8]. Mobile carriers using middleboxes to impose aggressive timeout value for idle TCP connections increase mobile devices battery consumption. Careless TCP middleboxes can facilitate certain network attacks, and even bring new attack vectors [4].

Furthermore, middleboxes have a negative impact on the TCP protocol, by hindering its evolution [9], [10]. They are likely to modify, filter, and drop packets that do not conform to their own policies, which can be over conservative, for example by suspiciously limiting the authorized features to a restricted subset. Generally speaking, we are witnessing the network infrastructure ossification. Alternative transport protocols that do not rely on TCP nor UDP, such as the Datagram Congestion Control Protocol (DCCP) [11] or the Stream Control Transmission Protocol (SCTP) [12], despite being standardized, fail to be deployed at large scale. The situation of the application layer is similar, with HTTP being the de-facto standard. TCP features also experience a myriad of tampering scenarios, which hampers TCP innovation initiatives [13].

The rationale behind such a blatant antagonism between innovation and network value is the reflect of that between the Internet stakeholders. Middleboxes are unilaterally deployed to fulfill manufacturers or network provider short-term commercial goals, while path transparency advocators have for only purpose to improve the Internet in the long-term [14], [15]. Because of this ongoing contention, researchers have to find devious way to produce innovation.

To overcome this, protocol designers have to ensure the middlebox-proofness of their solution. For example, recent discussions lead to the choice of UDP as a lightweight substrate for new protocols. Google's Quick Internet Connections (QUIC), currently used by Chrome browser, is a famous example of UDP-based protocol. It incorporates a multiplexed stream transport over UDP and its own application-level transport [16]. The design of the MultiPath TCP (MPTCP) feature, also required dedicated efforts to consider all possible in-path tampering and avoid unforeseen middlebox impairments [10].

In this paper, we study the transport-layer ossification and propose a simple intermediate classification of its factors. Based on a dataset collected from a large-scale campaign of active probing towards the most popular HTTP servers, we extract observations of in-path packet manipulations, we process the obtained observations to highlight the responsible middlebox policies and the path condition that they engender,

and we categorize middlebox-related impairments based on the potential negative consequences that they create on TCP traffic. Then, we use the resulting classes to give insights on the deployment and prevalence of path-impairing middleboxes in the wild. In particular, we show that a substantial percentage of network path are affected by feature or protocol-breaking middleboxes and that they are in majority located in the edge networks. We advocate for protocol designers to include a fallback mechanism carefully designed to ensure robustness to the classes of middleboxes described in this paper.

The remainder of this paper is organized as follows: Sec. II provides the required background on middlebox-related transport impairments, on the classification of middlebox policies, and on the algorithms we used to observe and analyze data; Sec. III details the active measurement campaign, and the analysis of gathered observations; finally, Sec. V concludes this paper by summarizing its main achievements.

II. Background

This section aims at providing the required background to understand the technical causes of the transport ossification. It shows examples of middlebox-related transport impairments, discusses the categories of impairing middlebox behaviors based on the *potential path condition* that they may cause, and introduces `tracebox` [17], the active middlebox detection algorithm that we used to build our dataset.

A. Impairments

Fig. 1 illustrates middlebox-induced transport impairments. In particular, Fig. 1a displays a typical impairment in which the TCP connection initiator wishes to negotiate the use of MultiPath TCP (MPTCP) [18], allowing a single TCP connection to simultaneously use multiple paths. To this end, it sends a `SYN` packet that carries the `MP_CAPABLE` option. A middlebox on the path to the destination is configured to drop all packets containing this option, leading so in a connectivity failure. Fortunately, MPTCP implements a fallback mechanism designed to cope with such policies, and the initiator retries to establish the connection with a regular TCP `SYN`. This prevents a complete connectivity failure, but increases the connection establishment latency.

In the example shown in Fig. 1b, the initiator attempts at negotiating the use of the Selective ACKnowledgment (SACK) option [19], by appending the dedicated `SACK_Permitted` option to the `SYN` packet. However, along the path, a middlebox strips the option and forwards the packet. The destination is SACK-capable and wishes to advertise it to the initiator by adding a `SACK_Permitted` option to the `SYN+ACK` packet, but the same middlebox strips it. Both endpoints attempted to negotiate the use of SACK, and failed.

In Fig. 1c, the connection initiator sends a TCP `SYN` packet that carries a Window Scale (WS) option [20], which is used to advertise windows larger than $2^{16}-1$, with a value of 8 that corresponds to a fixed factor of 2^8 to be applied to the target receive window. A middlebox, along the path, changes this value to 5, and the target stores this value. Reciprocally, the target sends the `SYN+ACK` with a WS of 11, which is then similarly modified. At the end of the TCP handshake, both endpoints have incorrect received window scaling factors.

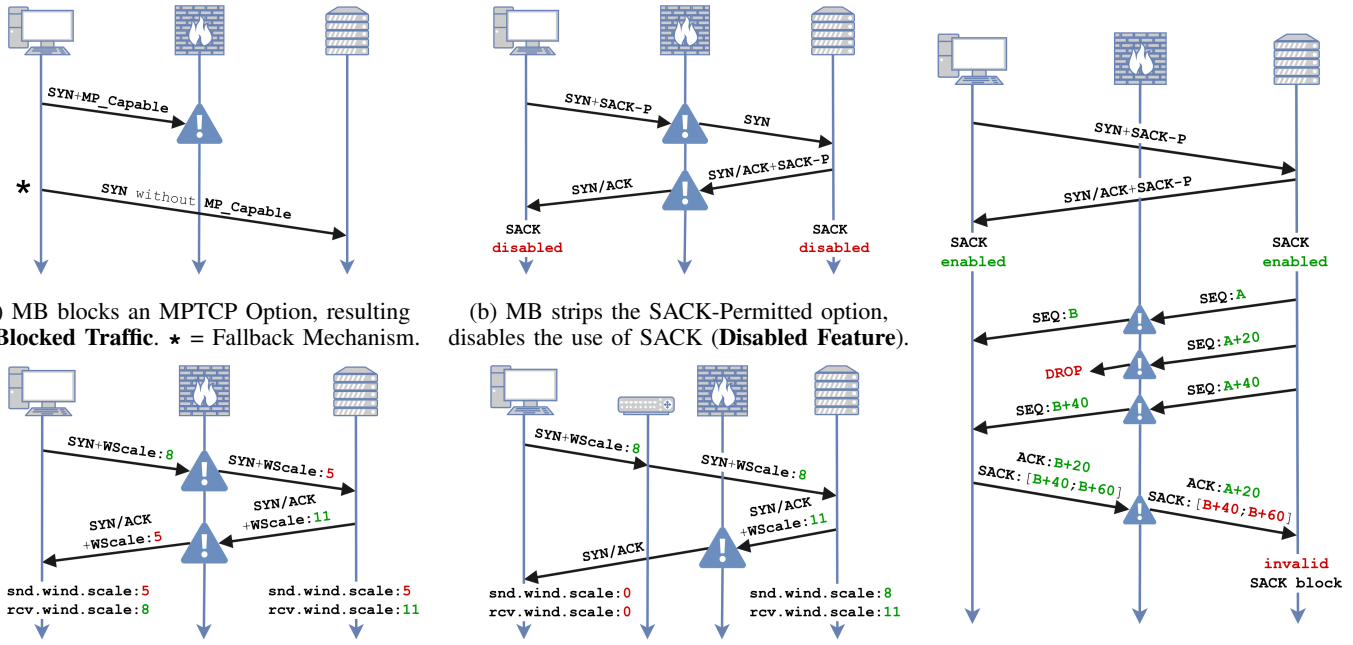
A second analogous example is shown in Fig. 1d in which the initiator sends a `SYN` packet with a WS of 8 that reaches the destination untouched. The target stores the value and sends back a `SYN+ACK` packet with a WS of 11. Due to asymmetric paths, the backward path crosses a middlebox that is not on the forward path and that strips the WS option. The initiator receives a `SYN+ACK` without WS, meaning that the other endpoint does not agree in using window scaling, and sets both factors value accordingly. The destination will then use window scaling, but the not the initiator.

Fig. 1e illustrates a scenario in which a middlebox applies modifications to packets that disrupt the TCP algorithm. The initiator wishes to negotiate the use of SACK and sends a `SYN` packet with `SACK_Permitted`. The destination agrees and sends back a `SYN+ACK` with `SACK_Permitted`, which results in enabling SACK for the connection. However, all exchanged packets exchanged cross a middlebox that performs a translation of TCP sequence numbers, in order to fix its lack of randomness and counter prediction attacks. The recipient sends three packet with 20 bytes of data, whose sequence numbers are translated from sequence space A to B, and the second packet experiences an unexpected drop. To notify the loss, the initiator acknowledges the first packet by setting `ACK` to $B + 20$, and the third with a SACK block. The `ACK` packets crosses back the middlebox, which translates the acknowledgment number, but unfortunately is not SACK-aware, and forwards to the target an invalid SACK block. Depending on its TCP implementation, the latter disregards the invalid SACK and needlessly retransmits the third packet, or discards the whole packet and stalls the connection.

B. Classification

We make an attempt at categorizing middlebox behaviors based on the *potential path condition* that they may cause, especially on transport protocols [21], [22], that aims at being used as an intermediate level of analysis between specific feature impairments and the global transport layer ossification phenomenon. To this end, we generalize the examples discussed in Sec. II-A.

The example of the MPTCP Option blocking shown in Fig. 1a is readily generalized to any other *path-dependent connectivity* issue. Indeed, we observe this behavior with, for a given path, ECN-setup `SYN` dropped in-path and non-ECN-setup `SYN` successfully reaching the destination. Similar events also exists with TCP Options other than MPTCP. Generally speaking, we notice a tendency of middleboxes to **block** unknown TCP features or transport protocols [23], [24], either explicitly (e.g., by sending TCP `RST` packets), or implicitly by dropping packets, to prevent the use of features considered unknown or unsafe. This category of impairment is symbolic of the transport-layer ossification because it has the most extreme consequences.



(a) MB blocks an MPTCP Option, resulting in **Blocked Traffic**. ⋆ = Fallback Mechanism.

(b) MB strips the SACK-Permitted option, disables the use of SACK (**Disabled Feature**).

(c) MB changes the WScale value (**Negotiation Disruption**).

(d) MB changes the WScale value and paths are asymmetric (**Negotiation Disruption**).

(e) A sequence number shuffling MB and a loss event result in **Disrupted Traffic**.

Fig. 1: Middlebox (MB) Impairments

In Fig. 1b, a TCP Option is stripped from both TCP `SYN` and `SYN+ACK` packets, resulting in the disabling of the `SACK` feature. This behavior is naturally generalized to any middlebox implementing **feature disabling** policies. For example, we observe multiple middleboxes stripping TCP options used for feature negotiation. In consequences, the feature in question cannot be used on the affected paths. Other variants, such as IP ECN bleaching middleboxes, that let the endpoints negotiate the use of ECN, but make the receiver unable to report congestion to the sender by systematically setting the IP ECN bits to zero. All these conditions fall within the same category of impairments, which is a softer version of the *blocked traffic* middlebox policies, that aims at normalizing traffic instead of blocking it.

Fig. 1c and Fig. 1d depict examples of middleboxes hindering the TCP window scaling factor announcement by rewriting the WScale option. We gather all path conditions that can potentially lead to **negotiation disruption** in the eponymous category. It includes occurrences of in-path changes of one-way state announcements. It also includes middleboxes with feature-disabling policies, which combined with an unfortunate load balancing or asymmetric paths, and in the absence of a resilient fallback mechanism, lead to inconsistent protocol states [10]. The latter is a subtle but direct consequence of the paradigm shift to `n`-way peering relationships, with protocols that still assume 2-way peering relationships [25].

Fig. 1e illustrates a scenario in which a connection is established, and both endpoints agrees in using the `SACK` option. A middlebox that re-shuffles TCP sequence number, and a loss event, leads to a bandwidth reduction. We generalize this scenario to all of in-path changes conditions that **disrupts transport control mechanisms** and result in performance reductions. For example, we observe cases of systematic changes of the IP ECN bits to 11 (i.e., Congestion Encountered), which results in substantial bandwidth reduction.

In summary, we categorize middleboxes based on the path conditions and the brokenness that they may spark, which can be a blocked traffic, disabled feature, negotiation disruption or traffic disruption.

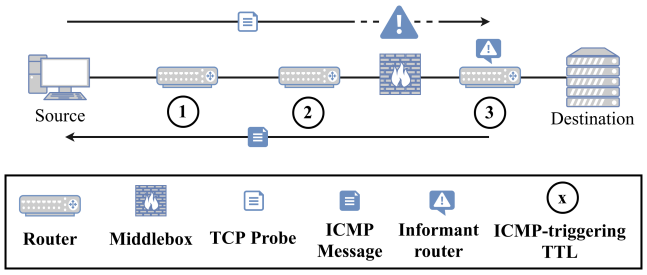


Fig. 2: `tracebox`

C. Detection

We used `tracebox` [17] to reveal the presence of middleboxes on a given path. Fig. 2 illustrates its operations.

`tracebox` probing mechanism is similar to `traceroute` [26], and works by sending TCP packets with incrementally increasing TTL values and by collecting the triggered answers. Then, it reads the payloads of the collected ICMP `time-exceeded` messages, which contains a copy of the expired probe received by the originator of the ICMP message, compares the value of each field of

the network and transport layer to the original probe, and infer in-path changes. Moreover, the TTL value of the ICMP message that first highlighted a modification gives a hint on the middlebox location on the path. By forging probes while varying the value of chosen fields and including different combinations of TCP Options, analyzing the sequences of triggered ICMP messages, `tracebox` is able to detect and locate middleboxes.

However, RFC792 [27] and RFC1812 [28] recommends ICMP messages to size the expired packet quotation differently, respectively, the entire IP header plus the first 64 bits of the transport header, and the entire headers of the expired packet. When a change is detected on a field located inside RFC792 quoting range, `tracebox` locates it between the router that first noticed the change, the *informant* router, and the router preceding it. When the changed field is located outside RFC792 quoting range, it introduces an *uncertainty zone*, between the informant router, and the closest RFC1812-compliant ingress router. We explain how we address this in Sec. III-A.

We point out that our method cannot detect all middleboxes along a path [17], and our results should, then, be considered as a lower bound.

III. Measurements

This section focuses on the quantification of middleboxes path impairments. In particular, Sec. III-A details our data collection process through large-scale `tracebox` probing, and Sec. III-C confront the middlebox classification introduced in Sec. II-B to the obtained dataset.

A. Data Collection & Processing

We collected our dataset by running `tracebox` on wired IPv4 networks, from 89 PlanetLab nodes located in North America (49), Europe (18), Asia (13), Oceania (6), and South America (3), each probing towards 594,241 HTTP servers, extracted from Alexa 1M list by selecting unique reachable addresses. The probes are TCP `SYN` packets designed to trigger as many conditional middlebox policies as possible, by varying the destination port (80, 8080, 8000, 8800, 443, 53, 12345, 1228, 34567)[2], the set of widely used and impairment-prone TCP Options (Maximum Segment Size, SACK-Permitted, Window Scale and MultiPath TCP)[3], over a three months period. This aims at maximizing the detection of drop and rewrite middlebox policies, which we analyze to highlight if they are likely to cause network dysfunctions.

The resulting dataset consists in all received packets from 948,457 unique IP addresses located in 2,977 different ASes, registered under 189 different country codes, from which we extracted 550 millions observations of middlebox-induced changes. In order to map observations to actual middleboxes and to reduce the location uncertainty, additional processing is applied to the entire set of observation. First, a label is assigned to each observations. When there is no location uncertainty, the label is the IP address of the router preceding the informant router. Otherwise, the label is assigned using heuristics and cross-checking between observations [3]. Second, observations are aggregated based on their label to build middlebox profiles. Third, we merge the obtained label-identified middlebox profiles by correlating the set of next hops, the observed changes, the quoting range, the sub-networks, and by performing alias resolution, into sets of observations that now consider as middleboxes. The resulting dataset consists in 232 millions observations attributed to 18,667 middleboxes in 372 ASes, which accounts for 2% of all visible network devices. Observations are discussed in Sec. III-B.

Then, the observation dataset is processed to extract path conditions. For a given path, defined by a source and a destination address, if all probes carrying a given option or feature (i.e., MPTCP, SACK-Permitted or ECN-setup) never reach past a given node, and at least one probe without the

Conditions	Observations	MBs	Consequences			
			BT	DF	ND	DT
Benign						
`dscp.changed`	143,548,746	7,227	✗	✗	✗	✗
`tcp.opt.mss.changed`	30,691,842	5,034	✗	✗	✗	✗
`ip.id.changed`	376,347	261	✗	✗	✗	✗
`ip.flags.changed.10`	6,312	6	✗	✗	✗	✗
`tcp.urg.changed`	954	1	✗	✗	✗	✗
`tcp.reserved.changed`	861	1	✗	✗	✗	✗
Inconclusive						
`tcp.checksum.changed`	34,101,880	11,276	✗	?	?	?
`ip.length.changed`	366,924	466	✗	?	✗	✗
`tcp.offset.changed`	29,069	32	✗	?	✗	✗
Impairments						
`tcp.seqnum.changed`[1]	17,745,019	211	✗	✗	✗	✓
`tcp.opt.mptcp.removed`	2,967,720	195	✗	✓	✗	✗
`tcp.opt.sackok.removed`	2,271,380	188	✗	✓	✓	✗
`tcp.opt.ws.changed`	82,811	49	✗	✗	✓	✓
`tcp.opt.ws.removed`	40,959	39	✗	✓	✗	✗
`tcp.opt.mss.removed`	31,841	31	✗	✓	✗	✗
`tcp.window.changed`	23,719	33	✗	✗	✗	✓
`ip.ecn.changed.00`	10,120	11	✗	✓	✗	✗
`tcp.ecn.changed.00`	6,507	6	✗	✓	✗	✗
`ip.ecn.changed.10`	7,270	6	✗	✓	✗	✗
`tcp.opt.mptcp.blocked`	3,171	6	✓	✓	✗	✗
`tcp.ecn.blocked`	2,646	6	✓	✓	✗	✗
`ip.ecn.changed.01`	1,011	4	✗	✓	✗	✗
`ip.ecn.changed.11`	544	4	✗	✗	✗	✓

TABLE I: Middlebox Impairments Overview. BT = Blocked Traffic. DF = Disabled Feature. ND = Negotiation Disruption. DT = Disrupted Traffic.

[1]This path condition is re-categorized as benign for certain analysis (See Sec. III-C).

[2]Each port is used for an entire campaign, except 80 and 443 which are used for three campaigns each.

[3]All campaigns are ECN-setup `SYN` probes with the same set of TCP Options (MSS, SACK-P and WScale), except for destination ports 80 and 443, that both run an additional campaign with an extra `MP_Capable` option, and another with a non-ECN-setup `SYN` probe without `MP_Capable`. The DSCP and IP-ID fields are initialized to 0, and the MSS to 1460.

feature does, it is a `blocked` condition. The answering node located right after the last answering node is considered to be the middlebox. `removed` refers to an option stripped from a TCP segment in the path, and `changed` refers to an observation of packet modification. We report the value assigned by middleboxes for a given field by appending it to the condition (e.g., `ip.ecn.changed.00`). For the sake of readability, initial and new value are not displayed in the table for all fields, but they are treated differently. In case of intermittent middlebox behavior, the most prevalent one is considered.

B. Observations

The obtained path conditions are summarized in Table I, divided between (i) *benign* observations of middlebox behavior that do not break the control plane, either because it is applied to fields designed for cooperation with middleboxes (i.e., DSCP) or routers, or because no negative impact on transport protocol was noticed (i.e., TCP MSS), (ii) *inconclusive* observations that are necessarily linked to in-path traffic manipulation but that can be linked to both benign and path-breaking middlebox policies, and finally (iii), observations of middlebox *impairments*.

Overall, we find that 75% (174.6 millions) of collected observations are linked to a benign behavior. The most frequent (143.5 millions) being the in-path change of Differentiated Services Code Point (DSCP), a 6-bit field in the IP header specially designed for classifying and marking network traffic in order to provide flow-level QoS. The second most frequent (30.5 millions) benign observation is a modification of the TCP Maximum Segment Size (MSS), a TCP Option field used to specify that maximum amount of data that be sent in a TCP packet. This behavior is not linked to any transport protocol problem. Finally, we observe changes of the IP-ID field, of the IP flags being set to `dont-fragment`, and of the TCP reserved field. Although this should not be performed by any in-path device, it has no significant impact on transport protocols.

Next, we find that 15% (34.5 millions) of observations cannot lead to a definite differentiation between benign and path-impairing policy. The most frequent one (34.1 millions) is the TCP checksum update, which can be a side effect of a benign policy (e.g., MSS change) or any other TCP modification, including feature-breaking's. We also observe changes of the IP total length field (0.3 millions), used to define the size of the entire packet, and of the TCP data offset field (29K), used to specify the size of the TCP header in 32-bit words. In the case of a TCP `SYN` segment, both form a strong indication of a TCP header size changed linked to a TCP option removed or added. The most set values for the IP total length field are 48 and 52, and for the TCP offset field are 7 and 8, which corresponds to a TCP `SYN` packet with 8 and 12 bytes of options. The discrepancy between the amount of observations of the two fields is linked to the higher visibility of the IP header to `tracebox` than latter TCP fields (See Sec. II-C). When taken alone, these observations are inconclusive.

Finally, we find that 10% (23.2 millions) of collected observations are linked to in-path middlebox packet manipulations that have the potential to harm transport protocol. The most frequent (17.7 millions) is the observation of the TCP sequence number randomizing box, which is supposed to fix an old vulnerability to a TCP sequence prediction attacks. However, it enables a similar TCP sequence number inference attack [29]. Moreover, as explained in Sec. II-A, when the same box fails to ensure consistency with semantically related fields (e.g., SACK blocks), it creates an inconsistency that can lead to a blackhole. The two next most frequent observations are those of the systematic removal of certain TCP Options (i.e., 2.9 millions MPTCP and 2.2 millions SACK-P), which is characteristic of the transport layer ossification. We also observe less frequent occurrences of MSS and WScale removals. However, there is a large discrepancy between the observed change of packet sizes via IP total length and TCP offset fields and the observations of options removal, which are a lot more frequent. This is explained by the widespread practice of removing TCP Options by overwriting them with padding bytes (NO-OPeration) without actually shrinking the packets, which require a costly copy operation.

We also observe in-path manipulation of window scaling factor, as well as the actual TCP window field, which has direct consequences on the amount of bytes exchanged and therefore on the QoS. Moreover, we observe rare occurrences of feature-dependent connectivity, where packets are dropped because they are carrying an MPTCP option (3,171), or because they try to negotiate ECN (2,646). Finally, we observe rare events of IP ECN manipulation that can lead to either the impossibility to use ECN to report any congestion (i.e., `ip.ecn.changed.00`, `ip.ecn.changed.01` and `ip.ecn.changed.10`), or to a systematic report of congestion (i.e., `ip.ecn.changed.11`).

Anecdotally, we found rare occurrences of TCP Options (i.e., MSS) being removed and added back by different devices on the same path.

Overall, we find that 18,230 middleboxes among the 18,667 that we detected are benign middleboxes, and that the remaining 437 are linked to path brokenness behavior. Among those 437 middleboxes, 211 are exclusively performing actions related to sequence number re-shuffling (i.e., `tcp.seqnum.changed`), while 226 are linked to other impairments.

C. Results

Then, we analyze the prevalence at the path level of each types of policies. It is different than observations and middleboxes count as it also takes into account their popularity, and reflects the chances of being subject to path impairments, for a user randomly chosen between the vantage points reaching any of our probing destination address at random. If there is more than one middlebox observation for the same path, we merge them as follows. If a blocked traffic impairment is observed,

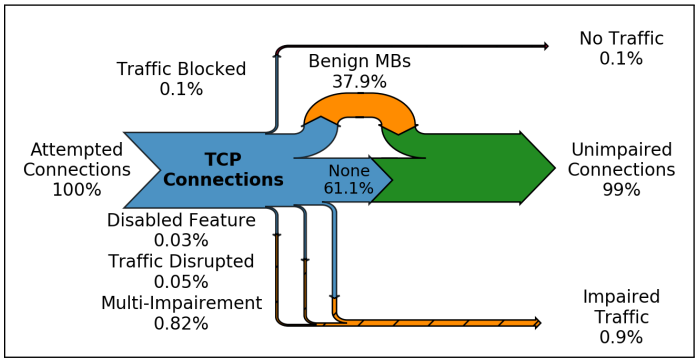


Fig. 3: Proportion of paths affected by potential middlebox impairments.

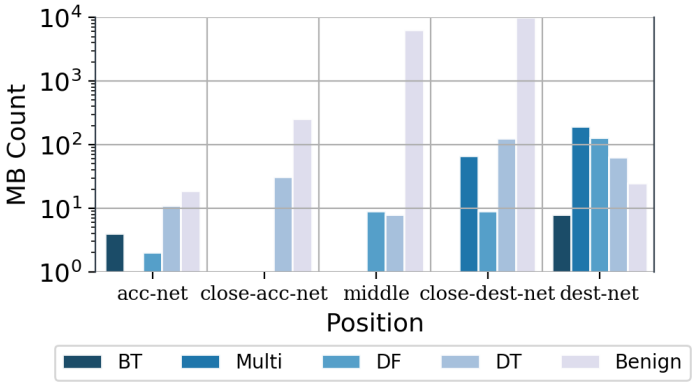


Fig. 4: Position of Middleboxes on the path. BT = Blocked Traffic. DF = Disabled Feature. ND = Negotiation Disruption. DT = Disrupted Traffic. Multi = Multiple Impairments.

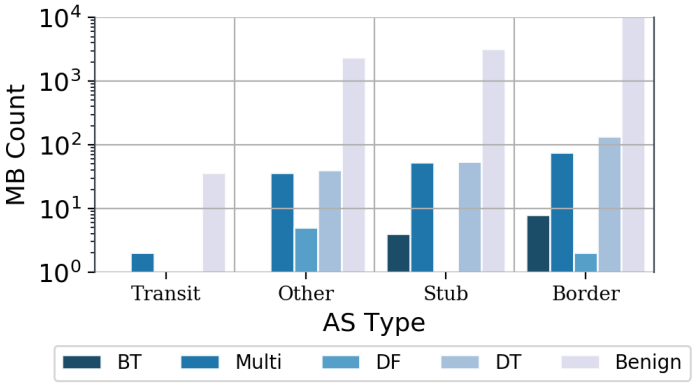


Fig. 5: AS types of Middleboxes on the path. Border = border routers between ASes of *different types*.

all other observations for the same path are discarded. Benign policies are discarded in favor of impairments, and if multiple impairments exists on the same path, we mark them accordingly. Finally, inconclusive policies are accounted as benign. The proportion of paths affected by each path condition is shown in Fig. 3.

Among the 52.8 millions paths probed by our algorithm, our detection method did not find any evidence of middleboxes presence in 32.3 millions (61.1%) of them, but it did discover that 20.5 millions (38.9%) paths are crossing at least one middlebox. More precisely, 32.4% of the paths include a benign middlebox, 6.5% a potential impairment. 0.1% of the paths involve a middlebox that blocks traffic, which in the absence of a fallback mechanism, results in a connectivity failure. 0.8% of paths are particularly broken, as they include multiple impairments (i.e., two or more among disabled feature, traffic disruption and negotiation disruption). Finally, 5.6% of paths are affected by traffic disruption middleboxes. The majority (5.5%) of the policies of this last category requires a rare combination of factors to actually impair traffic, using SACK, a broken SACK policy, and a loss event, (See Sec. II-A) and for that reason and to avoid giving an unbalanced perception of middlebox impairments, we choose to re-categorize those policies as benign. However, it should be noted that new features that wishes to include TCP sequence numbers elsewhere than in the dedicated fields should address them, as they still hold traffic disruption potential. For example, MPTCP make use of a data sequence mapping scheme that specifies a mapping from each sub-flow sequence space to the global data sequence space, in terms of starting sequence numbers and length of the mapping validity [30].

In short, 38.9% of network paths involve at least one middlebox, and 1% are affected by transport impairments other than `tcp.seqnum.changed`. At first glance, the ratio of paths affected by transport-breaking conditions might seem relatively low, but it is in fact, without proper fallback mechanisms built to address each classes of middlebox impairments (e.g., MPTCP fallback mechanism [31]), largely sufficient to hamper protocols or feature until disappearance. Moreover, we observe a tendency of feature-breaking middlebox to affect the same paths, either with broken middleboxes combining multiple path impairments, or with multiple middleboxes affecting the same path.

It should also be noted that 5,924 middleboxes among the 18,667 in total (32%) are exclusively linked to observations of TCP checksum update, which we are forced to classify as inconclusive. Still, these middleboxes are necessarily performing other changes on the TCP header, either benign (e.g., `tcp.opt.mss.changed`), or impairing. The middlebox is invisible either because it is required to, for example NATs are invisible for endpoints to be able to map ICMP messages to the right sockets [32], or because they chose to evade `tracebox` detection. This consolidates the assertion that all of our findings should be treated as lower bounds.

Then, we computed the position of middleboxes by joining IP addresses into sub-networks, removing the access and destination networks from the obtained network list, and splitting the rest in three sections. The first set of networks is marked as *close to the access network*, the second as *middle*, and the last as *close to the destination network*. The results are displayed in Fig. 4. We also show the AS types, resolved from BGP Routing Information Base (RIB) data, in Fig. 5.

First, it shows that most benign middleboxes are located in the middle of the path, or close to the destination network.

More importantly, it shows that blocked traffic is exclusively the action of broken middleboxes located in access networks or destination networks. Then, it reveals that multi-impairments middleboxes are located at the destination network or at the network right before, and that disabled feature and disrupted traffic follow the same tendency. Most impairments are located on the destination network or close to it, and fewer are located on client-side access networks. Fig. 5 shows that almost no impairments are found in transit ASes. Moreover, most impairments are located in stub networks or at the border of different types of ASes.

Overall, we observe a clear inclination of the most dangerous middleboxes to be located in the edge networks.

IV. Related Work

Multiple active measurements algorithms designed to reveal the presence of middleboxes and characterize their behavior exists. `tbit` [33] was an early attempt at studying the interactions between transport protocols and middleboxes. It works by sending TCP packets to a server while varying the ECN value, IP, and TCP Options of the probe, and analyses exclusively the TCP answers. It is able to detect feature-dependent connectivity and a few middlebox tampering. However, `tbit` is not able to locate middleboxes.

`TCPExposure` [9] is a client/server application exchanging specially crafted packets to detect middleboxes interference on the path. The client opens a raw socket and uses it to send TCP packets towards the server. The server answers by encapsulating the received packets with a copy of its own packets. Upon reception, the client is able to infer middlebox changes in both directions. However, `TCPExposure` requires to control both ends of the path, making it unusable for a middlebox census.

TCP HICCUPS [34] is a TCP extension that is able to perform in-band detection of middleboxes tampering with packet headers. It hashes sensitive fields and stores the results in three supposedly unused TCP fields. However, it also requires control on both endpoints, with capabilities of updating the TCP/IP stack.

`ECN Spider` [35] is a TCP client that checks for ECN-related problems on the path to a given destination. By sequentially opening two TCP connections with ECN and without, it is able to detect occurrences of path-dependent connectivity and ECN signaling anomalies.

Tools focusing on connectivity issues also exist. For instance, `Netalyzr` [8] can be used to perform A/B testing reachability tests with selected transport protocols and destinations ports. More recently, `PATHSpider` [36] was introduced and works similarly.

`copycat` [24] is a transport protocol testing tool that generates flows mimicking the TCP behavior with the wire image of another transport protocol. It is able to highlight any differential treatment between TCP and the protocol under test, in term of connectivity and QoS.

Hesmans and al. used MBTest [10], a minimal `Click`-based middlebox, to experimentally evaluate how it interacts with the Linux TCP stack. The study focuses on TCP Options impairments, and concludes that endpoints should not assume that transport and network headers will not be modified on the path, and explains how MultiPath TCP has been designed to be middlebox-proof.

These tools provide great results, but they are limited to specific paths as both ends of the path must be under control or must implement particular techniques in the TCP/IP stack.

Finally, the RFC3234 [1] establishes a catalogue of middleboxes, and proposed to classify them according to eight functional characteristics. It briefly addresses middlebox-related impairments, and acknowledges the expiring of the the end-to-end paradigm.

V. Conclusion

For many years, network actors have been struggling for important architectural decisions. Different parties with divergent concerns, increasing value of the middle network versus enabling end-to-end innovation, have been colliding. On the one hand, Internet researchers hold a long-term vision of the Internet, driven by the sole universal goal of enhancing it. On the other hand, middlebox vendors and network providers wish to fulfill narrower short-term interests, such as commercial, surveillance, or increased control on network traffic. This lack of cooperation between the different parties not only preclude from theoretically achievable architectural purity, but might also force one party to workaround technical decisions of the other (e.g., NAT traversal mechanisms), and even cripple innovation, by introducing a phenomenon of transport layer ossification.

In this paper, we presented a bottom-up study of this phenomenon. We investigated diverse low-level middlebox-related transport impairments, and proposed an intermediate reading grid for better understanding of the Internet-level dynamics.

First, we proposed a classification of middlebox impairments, that categorizes middlebox-related impairments based on the potential negative consequences that they create on TCP traffic, to be used as an intermediate level of analysis between specific feature impairments and the global transport layer ossification phenomenon.

Then, we conducted a large-scale active probing campaign towards the most popular HTTP servers, with the help of a measurement tool (i.e., `tracebox`) that allows for detection and location of middleboxes along a path, while only requiring control on a single endpoint, and collected a dataset composed of more than half a billion observations of in-path packet manipulation. We extracted middleboxes from the obtained observations, highlighted the responsible policies and the path condition that they engender, for regular TCP traffic with or without new features.

Finally, we used our classes of middlebox-induced path brokenness to characterize path-impairing middleboxes in the wild, by quantifying their deployment, prevalence, and positioning. Briefly, we showed that (i) at least 2% of deployed network devices are TCP/IP middleboxes, that (ii) more than

one third of network paths are crossing at least one of them, that (iii) a substantial part, at least 437 middleboxes covering 6.5% of all paths, are harming TCP traffic, forbidding innovation, and participating in the transport ossification, and that (iv) the majority of the dangerous middleboxes are located in edge networks.

Consequently, we advocated to protocol designers for including carefully designed fallback mechanisms to ensure robustness to each of the middlebox classes described in this paper, which despite not being ideal (i.e., extra latency), prevents from more serious failures.

Overall, we provided an intermediate-level analysis of the transport-layer ossification of the network infrastructure. We achieved this by establishing a classification of middlebox-induced path conditions that can be used as a guideline when developing new protocols or features, to fill the gap between fine-grained transport impairments and the transport ossification global phenomenon, and showed its extent by confronting it the Internet.

ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 688421. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information.

REFERENCES

[1] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and issues," Internet Engineering Task Force, RFC 3234, February 2002.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. ACM SIGCOMM*, August 2012.

[3] K. Edeline and B. Donnet, "A first look at the prevalence and persistence of middleboxes in the wild," in *Proc. International Teletraffic Congress (ITC)*, September 2017.

[4] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proc. ACM SIGCOMM*, August 2011.

[5] A. Lutu, M. Bagnulo, A. Dhamdhere, and k. claffy, "NAT revelio: Detecting NAT444 in the ISP," in *Proc. Passive and Active Measurement Conference (PAM)*, March/April 2016.

[6] L. D'Acunto, N. Chiluka, T. Vinò, and H. J. Sips, "Bittorrent-like P2P approaches for VoD: a comparative study," *Computer Networks (COMNET)*, vol. 57, no. 5, pp. 1253–1276, April 2013.

[7] T. Barbette, C. Soldani, and L. Mathy, "Fast userspace packet processing," in *Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, July 2015.

[8] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

[9] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2011.

[10] B. Hesmans, F. Duchene, C. Paasch, G. Detal, and O. Bonaventure, "Are TCP extensions middlebox-proof?" in *Proc. Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox)*, December 2013.

[11] E. Kohler and S. Handley, M. an Floyd, "Datagram congestion control protocol (DCCP)," Internet Engineering Task Force, RFC 4340, March 2006.

[12] R. Stewart, "Stream control transmission protocol," Internet Engineering Task Force, RFC 4960, September 2007.

[13] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tüxen, M. Welzl, D. Damjanovic, and S. Mangiante, "De-ossifying the Internet transport layer: A survey and future perspectives," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 619–639, 2017.

[14] C. Huitema, "The secure transport tussle," in *IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)*, January 2015.

[15] D. Clark, J. Wroclawski, K. Sollins, and R. Braden, "Tussle in cyberspace: Defining tomorrow's Internet," *IEEE/ACM Transactions on Networking (ToN)*, vol. 13, no. 3, pp. 462–475, June 2005.

[16] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC transport protocol: Design and Internet-scale deployment," in *Proc. ACM SIGCOMM*, August 2017.

[17] G. Detal, b. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2013.

[18] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," Internet Engineering Task Force, RFC 6824, January 2013.

[19] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," Internet Engineering Task Force, RFC 2018, October 1996.

[20] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, "TCP extensions for high performance," Internet Engineering Task Force, RFC 7323, September 2014.

[21] K. Edeline and B. Donnet, "Towards a middlebox policy taxonomy: Path impairments," in *Proc. 7th IEEE International Workshop on Science for Communication Networks (NetSciCom)*, April 2015.

[22] K. Edeline and B. Donnet, "On a middlebox classification," in *IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)*, January 2015.

[23] K. Edeline, M. Kühlewind, B. Trammell, E. Aben, and B. Donnet, "Using UDP for Internet transport evolution," arXiv, cs.NI 1612.07816, December 2016.

[24] K. Edeline, M. Kühlewind, B. Trammell, and B. Donnet, "copycat: Testing differential treatment of new transport protocols in the wild," in *Proc. ACM/IRTF/ISOC Applied Networking Research Workshop (ANRW)*, July 2017.

[25] M. A. Lemley and L. Lessig, "The end of end-to-end: Preserving the architecture of the Internet in the broadband era," University of California at Los Angeles, Technical Report 2000-19, October 2000.

[26] V. Jacobson et al., "traceroute," UNIX," man page, 1989.

[27] J. Postel, "Internet control message protocol," Internet Engineering Task Force, RFC 792, September 1981.

[28] F. Baker, "Requirements for IP version," Internet Engineering Task Force, RFC 1812, June 1995.

[29] Z. Qian and Z. M. Mao, "Off-path tcp sequence number inference attack-how firewall middleboxes reduce security," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 347–361.

[30] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," Internet Engineering Task Force," RFC, 2013.

[31] M. Scharf and A. Ford, "Multipath TCP (MPTCP) application interface considerations," Internet Engineering Task Force, RFC 6897, March 2013.

[32] S. Guha, B. Ford, S. Senthil, and S. Pyda, "NAT behavioral requirements for ICMP," Internet Engineering Task Force, RFC 5508, April 2009.

[33] A. Medina, M. Allman, and S. Floyd, "Measuring interactions between transport protocols and middleboxes," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2004.

[34] R. Craven, R. Beverly, and M. Allman, "Middlebox-cooperative TCP for a non end-to-end Internet," in *Proc. ACM SIGCOMM*, August 2014.

[35] B. Trammell, M. Kühlewind, D. Boppart, L. I., G. Fairhust, and R. Scheffenegger, "Enabling Internet-wide deployment of explicit congestion notification," in *Proc. Passive and Active Measurement Conference (PAM)*, March/April 2015.

[36] I. R. Learmonth, B. Trammell, M. Kühlewind, and G. Fairhurst, "PATHspider: A tool for active measurement of path transparency," in *Proc. ACM/IRTF/ISOC Applied Networking Research Workshop (ANRW)*, July 2016.