IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Jiancheng Guo et al.	§	Attorney Docket No.: U022-0007RE
	§	
U.S. Patent 9,578,040	§	
	§	
Issue Date: February 21, 2017	§	Customer No.: 29,150
	§	
Filing Date: December 16, 2014	§	
	§	
For: PACKET RECEIVING METHOD,	§	
DEEP PACKET INSPECTION	§	
DEVICE AND SYSTEM	§	

Mail Stop "*Ex Parte* Reexam"
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REQUEST FOR *EX PARTE* REEXAMINATION OF U.S. PATENT 9,578,040

Dear Commissioner:

Pursuant to the provisions of 35 U.S.C. §§ 301-307, Unified Patents, LLC ("Requester") hereby requests an *ex parte* reexamination of claims 1, 4-6, and 9-11 (the "Challenged Claims") of U.S. Patent 9,578,040 (the "'040 Patent", Ex. 1001), which issued on February 21, 2017 to Jiancheng Guo et al. from U.S. Patent Application 14/572,514 (the "''514 Application"), filed on December 16, 2014 and claiming priority to International Application PCT/CN2021/077994 filed June 30, 2012.[1] The '040 Patent is currently purportedly assigned to Orbit Licensing LLC ("Orbit" or "Patent Owner"), according to the assignment record recorded in the U.S. Patent and Trademark Office ("USPTO") at reel/frame 054442/0337.

Requester submits that this Request presents prior art references and analyses that are

[1] At this time, for purposes of this Request, Requester assumes the '040 Patent is entitled to its claimed priority date of June 30, 2012.

noncumulative of the prior art before the Examiner during original prosecution of the '040 Patent, and that the Challenged Claims are invalid over these references. Requester therefore requests that an order for reexamination and an Office Action rejecting claims 1, 4-6, and 9-11 be issued.

Ex Parte Patent Reexamination Filing Requirements

Pursuant to 37 C.F.R. § 1.510(b)(l), statements pointing out at least one substantial new question of patentability based on material, non-cumulative reference patents and printed publications for the Challenged Claims of the '040 Patent are provided in Section II of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(2), reexamination of the Challenged Claims of the '040 Patent is requested, and a detailed explanation of the pertinence and manner of applying the cited references to the Challenged Claims is provided in Section III of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(3), copies of every patent or printed publication relied upon or referred to in the statement pointing out each substantial new question of patentability or in the detailed explanation of the pertinence and manner of applying the cited references are provided as Exhibits 1005 to 1011 of this Request.

Pursuant to 37 C.F.R. §1.510(b)(4), a copy of the '040 Patent is provided as Exhibit 1001 of this Request, along with a copy of any disclaimer, certificate of correction, and reexamination certificate issued corresponding to the patent.

Pursuant to 37 C.F.R. § 1.510(b)(5), the attached Certificate of Service indicates that a copy of this Request, in its entirety, has been served on Patent Owner at the following address of record for Patent Owner, in accordance with 37 C.F.R. § 1.33(c):

Gerald T. Gray
Leydig, Voit & Mayer, Ltd.
(for Huawei Technologies Co., Ltd)
Two Prudential Plaza Suite 4900
180 North Stetson Avenue
Chicago IL 60601

Also submitted herewith is the fee set forth in 37 C.F.R. § 1.20(c)(l).

Pursuant to 37 C.F.R. § 1.510(b)(6), Requester hereby certifies that the statutory estoppel provisions of 35 U.S.C. § 315(e)(l) and 35 U.S.C. § 325(e)(l) do not prohibit Requester from filing this *ex parte* patent reexamination request.

TABLE OF CONTENTS

TABLE OF EXHIBITS

Exhibit	**Description**
Ex. 1001	U.S. Patent 9,587,040 to Guo et al. ("the '040 patent")
Ex. 1002	File History of the '040 Patent
Ex. 1003	Declaration of Angelos Keromytis, Ph.D.
Ex. 1004	*Curriculum Vitae* of Angelos Keromytis, Ph.D.
Ex. 1005	Leech et al., "SOCKS Protocol Version 5," RFC 1928, March 1996 ("RFC 1928")
Ex. 1006	Koblas et al., "SOCKS," 1992, UNIX Security Symposium III Proceedings, 1992 ("Koblas")
Ex. 1007	Kitamura, "A SOCKS-based IPv6/IPv4 Gateway Mechanism," RFC 3089, April 2001 ("RFC 3089")
Ex. 1008	U.S. Patent Application Publication No. 2009/0157889 ("Treuhaft")
Ex. 1009	U.S. Patent Application Publication No. 2006/0167871 ("Sorenson")
Ex. 1010	U.S. Patent Application Publication No. 2004/0006621 ("Bellinson")
Ex. 1011	Subramanian et al., "An empirical vulnerability remediation model, IEEE International Conference on Wireless Communications, Networking and Information Security, 2010 ("Subramanian")
Ex. 1012	Excerpts from Microsoft Computer Dictionary, Fifth Edition, 2002
Ex. 1013	U.S. Patent 6,950,660 to Hsu et al. ("Hsu")
Ex. 1014	U.S. Patent Application Publication No. 2009/0049539 to Halbedel et al. ("Halbedel")
Ex. 1015	Declaration of Ingrid Hsieh-Yee, Ph.D. (including Appendices A, B, and 1005A through 1011C)
Ex. 1016	Declaration of Duncan Hall of the Internet Archive regarding the public availability of RFC 1928, Koblas, RFC 3089, and Subramanian

I. SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

Before describing the substantial new questions of patentability presented in this Request, provided below is an overview of the '040 Patent, a discussion of claim construction, and a summary of the prior art being discussed in the present Request.

A. U.S. Patent 9,578,040

1. Brief Overview of the Domain Name System (DNS)

Certain aspects of the '040 Patent, and the prior art references discussed in this Request, touch on the Domain Name System (DNS). DNS is the hierarchical naming system in which computers, services, or other resources connected to the Internet have an address represented as both a human-readable domain name (e.g., www.google.com) and a machine-readable Internet Protocol (IP) address (e.g., 111.222.333.444). Ex. 1003, ¶¶ 27-29; Ex. 1012, 4. DNS has been a basic, essential component of the functionality of the Internet since the late 1980s.

At a high level, the DNS includes a collection of name servers that maintain the domain name hierarchy and provide translation services between the domain name and IP address spaces. Ex. 1003, ¶28. The name servers store address records mapping domain names to their corresponding IP addresses. *Id.* A client application—such as a web browser—needs an IP address to access a resource connected to the Internet—such as a server hosting a website. *Id.* Sometimes, however, the application does not have the particular IP address of the resource to which access is sought. *Id.* Instead, the application may only have the corresponding domain name of that resource, as when a user enters the domain name (e.g., www.google.com) into a web browser. *Id.*

To obtain the IP address of the Internet resource, the client application sends a DNS query to a name server in the DNS. *Id.*, ¶ 29. The DNS query includes the domain name of the resource to which the application seeks access and requests the name server to provide the corresponding IP address. *Id.* Upon receiving the DNS query, the name server "resolves," or converts, the domain into its corresponding IP address by looking up the address record for the domain name in the DNS query, obtaining the corresponding IP address from the address record, and returning the IP address to the client application in a DNS response. *Id.* As this point, the client application can now access the Internet resource directly using the "resolved" IP address. *Id.* This process of DNS resolution typically occurs transparently to the user of the client. *Id.*

2. Summary of the '040 Patent

The '040 Patent is directed to a "a packet receiving method, a deep packet inspection and system." '040 Patent (Ex. 1001), 1:14-16. The deep packet inspection (DPI) device receives a service request packet sent by a terminal device. *Id.*, 3:27-28, FIG. 1 (step S101). The service request packet contains two pieces of information: (1) a terminal domain name indicating the terminal device; and (2) a server domain name indicating a server requested by the service request. *Id.*, 3:28-31. Upon receiving this service request, the DPI device determines whether to discard the service request packet or establish the requested connection. *See id.*, 3:66-5:46. To do this, the DPI device first resolves (i.e., converts) the server domain name into its corresponding IP address, e.g., using DNS. *Id.* 3:66-4:12, FIG. 1 (step S102). Having resolved the IP address of the server to which the terminal device wants to connect, the DPI device now determines whether that IP address is contained in a preset list of allowable addresses for the terminal domain name. *Id.*, 4:13-5:47, FIG. 3 (step S103). If the IP address is not on the list, the DPI device discards or denies the service request, preventing the terminal device from accessing the IP address of the server. *Id.* But if the IP address is on the list, the DPI device establishes the requested connection. *Id.*

Copies of the '040 Patent and its file history are provided as Exhibits 1001 and 1002, respectively. For the sake of reference, the Challenged Claims (i.e, claims 1, 4-6, and 9-11), are reproduced below. Claims 1, 6, and 11 are the independent claims, while the remaining challenged claims depend directly or indirectly from these claims.

> 1. A packet receiving method, comprising:
>
> receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;
>
> resolving the received server domain name to obtain a service server Internet protocol (IP) address; and
>
> discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device.
>
> 4. The method according to claim 1, wherein, after the resolving the received server domain name to obtain the service server Internet protocol

(IP) address, the method further comprises:

if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device.

5. The method according to claim 1, wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:

if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device.

6. A deep packet inspection (DPI) device comprising a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising:

receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;

resolving the server domain name to obtain a service server Internet protocol (IP) address; and

discarding the packet if the service server IP address resolved does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with accessible service server IP addresses under an access authority of the terminal device.

9. The DPI device according to claim 6, wherein after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:

if the service server IP address resolved belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device.

10. The DPI device according to claim 6, wherein after the resolving the received server domain name to obtain the service server Internet protocol

(IP) address, the method further comprises:

if the service server IP address resolved belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device.

11. A system, comprising:

a deep packet inspection (DPI) device; and

a terminal device, configured to send a service request packet to the DPI device, wherein the packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request sent by the terminal device;

the DPI device having a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising:

receiving the service request packet sent by the terminal device;

resolving the server domain name received to obtain a service server Internet protocol (IP) address; and

discarding the packet if the service server IP address resolved does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with accessible service server IP addresses under an access authority of the terminal device.

In general, the claims of the ’040 Patent relate to how the DPI device determines whether to discard a service request packet received from a terminal device or to establish the connection based on two pieces of information contained in the request—a domain name of the terminal and a domain name of the server. Figure 6 of the ’040 patent, reproduced above, shows the system including the DPI device 30 and one or more terminal devices 40.

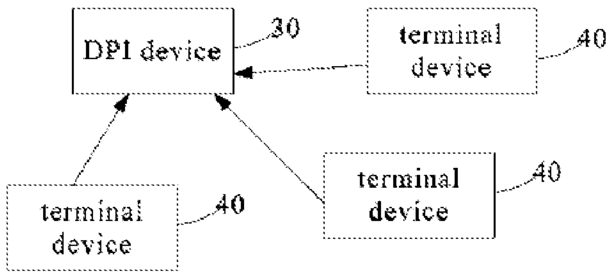


FIG. 6

Referring to the method shown in Figure 1 of the ’040 Patent, reproduced below, when a terminal device wants to connect to a server, it sends a service request packet to the DPI device. *Id.*, 3:27-28, 6:34-25, FIG. 3 (step S101); *see also id.*, 6:34-38, FIG. 4 (step S204). The service

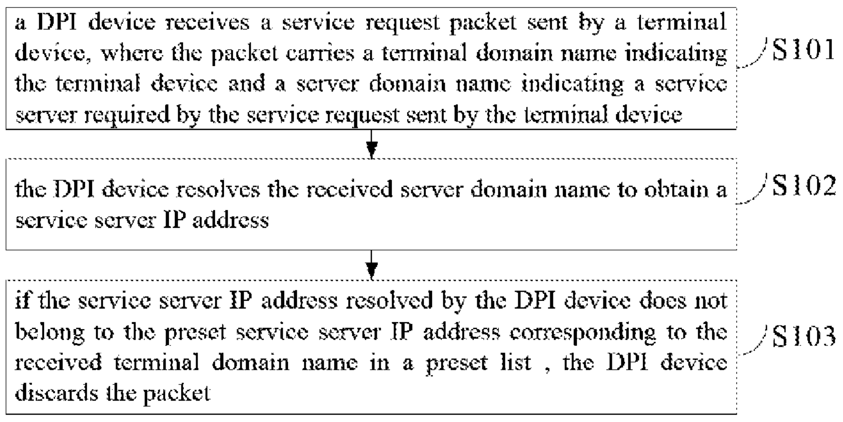

a DPI device receives a service request packet sent by a terminal device, where the packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request sent by the terminal device
S101
the DPI device resolves the received server domain name to obtain a service server IP address
S102
if the service server IP address resolved by the DPI device does not belong to the preset service server IP address corresponding to the received terminal domain name in a preset list , the DPI device discards the packet
S103

FIG. 1

request packet contains two pieces of information: (1) a terminal domain name indicating the terminal device; and (2) a server domain name indicating a server requested by the service request. *Id.*, 3:28-31; *see also id.*, 6:34-38. The terminal domain name is simply "a unique identifier" the terminal device uses to identify itself to the DPI device from among "tens of thousands of terminal devices and service servers in the network." *Id.*, 3:32-51. And, similarly, the server domain name is simply a unique identifier of the server that the terminal device wishes to connect with. *Id.*, 3:51-65.

Upon receiving the service request, the DPI device determines whether to discard it or establish the requested connection based on the two pieces of information in the request—the terminal domain name and server domain name. *See id.*, 3:66-5:4, 6:39-50. It is a two-step process:

First, the DPI device resolves (i.e., converts) the server domain name into its corresponding IP address. *Id.*, 3:66-4:12, FIG. 1 (step S102); *see also id.*, 6:39-41 (FIG. 3, step S205). For example, using DNS, the DPI device may convert the server domain name (e.g., www.google.com) to its corresponding machine-readable IP address (e.g., 2.2.2.2). *Id.*, 4:1-12.

Second, having resolved the IP address of the server to which the terminal device wants to connect, the DPI device now determines whether that IP address is contained in a preset list of accessible (i.e., allowable) addresses for the terminal domain name. *Id.*, 4:13-5:47, FIG. 3 (step S103); *see also id.*, 6:42-49, FIG. 3 (step S206). The "preset list is preset in the DPI device in advance" as configuration information before the method in Figure 1 occurs.[2] The '040 Patent

[2] In steps S201-203, the '040 Patent describes a process by which the DPI device establishes the preset list before the process in Figure 1, and steps S204-S206 of Figure 3, occur. *Id.*, 5:54-6:30, FIG. 3 (steps S201-S203). But establishing the preset list relates to unchallenged claims 2, 3, 7, and 8, so it is peripheral to this Request and not discussed in detail here.

gives an example of the preset list in Table 1 in the specification:

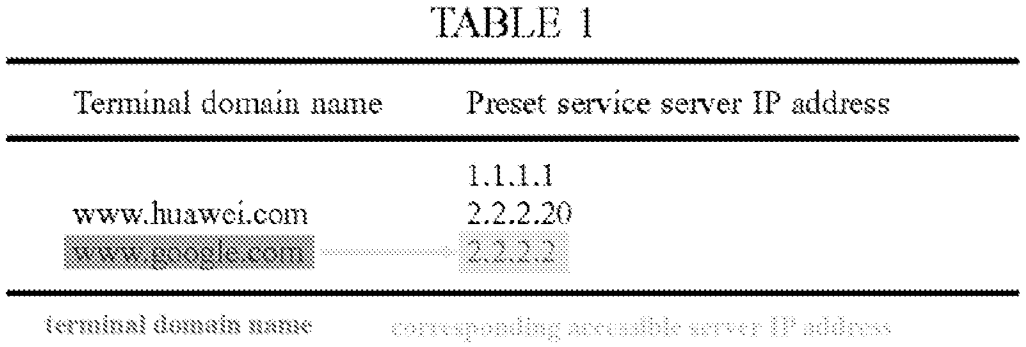


TABLE 1

Terminal domain name	Preset service server IP address
	1.1.1.1
www.huawei.com	2.2.2.20
www.google.com	2.2.2.2

terminal domain name
corresponding accessible server IP address

Ex. 1001, 4:27-34[3]. The left column lists terminal domain names of various terminal devices, and the right column lists the corresponding preset server IP addresses that the terminal domain names have authorization to access. *Id.* 4:35-5:24. For example, as highlighted above, a terminal device at the domain name www.google.com has authorization to access the corresponding server IP address 2.2.2.2 but not IP address 2.2.2.20. *Id.* A terminal device located at the terminal domain name www.huawei.com, on the other hand, may access the corresponding server IP address 2.2.2.20 but not 2.2.2.2. *Id.*

Continuing with step S103 of Figure 1, to determine whether to discard the IP address resolved from the server domain name contained in the request, the DPI device checks whether the preset list identifies the resolved IP address as an accessible IP address for the terminal domain name contained in the request.[4] Ex. 1001, 4:13-5:24; *see also id.*, 6:42-57, FIG. 3 (step S206). In

[3] Requester designates annotated figures with "*"

[4] Perhaps stemming from translation of its original Chinese text, the '040 patent uses somewhat strange language to describe the scenarios when the preset list does or does not list the resolved IP address as an accessible IP address for a terminal domain name. Rather than state that the IP address is or is not a preset IP address on the list, the '040 patent respectively states that the resolved IP "belongs" or "does not belong" to a preset IP address on the list. *See* Ex. 1001, Abstract, 2:1-4, 2:15-19, 2:36-38, 4:16-17, 5:5-9, 5:32-35, 6:42-53, 6:63-66, 7:64-67, 8:34-38, 8:47-52, 8:55-59, 9:35-38, 10:12-15. A POSA would have understood that, by saying "belongs" or "does not belong" to a preset address on the list, the '040 patent respectively means that the IP address is or is not listed as a preset address on the list. Ex. 1003, ¶ 38, n. 3.

Table 1 above, for example, if the request came from the terminal domain name www.google.com, the DPI device would check whether the resolved IP address is listed in the right column as a corresponding accessible IP address for the terminal domain name www.google.com. *Id.*

If the resolved IP address is not on the preset list, the DPI device discards the request, preventing the terminal device from accessing the server at the resolved IP address. *Id.*, 4:13-5:17, FIG. 1 (step S103); *see also id.*, 6:50-57, FIG. 3 (step S206). For example, if the terminal domain name is www.google.com and the resolved IP address is 2.2.2.20, which is listed an authorized IP address for the terminal domain www.huawei.com but not for www.google.com, the DPI device would discard the request. *Id.*, 4:36-5:24. But if the resolved IP address is on the list for the terminal domain name—as the IP address 2.2.2.2 for the terminal domain name www.google.com---the terminal device has authorization to connect to the server and thus the DPI device establishes the connection. *Id.*, 6:36-7:5, FIG. 3 (step S207); *see also id.*, 5:17-24.

According to the '040 Patent, its technique of checking whether the resolved IP address is on the preset list for the terminal domain name, before establishing the connection, helps prevent a terminal device from fraudulently connecting to server by later changing its domain name when it connects to the server. *See id.*, 6:51-57, 8:39-46. Specifically, the '040 patent identifies a purported problem in which a terminal device, after obtaining the IP address of a server it is not authorized to access, changes its domain name in the host field of a connection request it later sends to IP address. *Id.* According to the '040 Patent, servers typically do not check the host field of a connection request to make sure the terminal device is authorized, and thus cannot prevent an unauthorized connection if the terminal device changes its domain name after it has already obtained the server's IP address. *See id.*, 1:26-48, 4:58-67, 5:10-17.

The '040 Patent gives an example in which a terminal device with the domain name www.google.com gains free access to a charged website—which the terminal domain name www.huawei.com has authorization to access but the terminal domain name www.google.com does not—by changing its domain name from www.google.com to www.huawei.com after obtaining the IP address of the server hosting the charged website. *See id.*, 1:26-48, 4:35-67, 5:5-17, 8:39-46. As shown in Figure 2, after obtaining the resolved IP address of the charged website, the terminal device alters its domain name from www.google.com to www.huawei.com in the host field of an HTTP GET request it sends to the resolved IP address. *Id.* 4:43-67, FIG. 2. According to the ’040 Patent, by checking that the preset list contains the IP address of the server as authorized for the terminal domain name, the DPI device can discard the request before the terminal device obtains the IP address of the server and attempts an unauthorized connection. *See, e.g.*, 6:51-57, 8:39-46.

3. Prosecution History

The ’040 Patent issued from U.S. Application No. 14/572,514 (“the ’514 Application”) filed April 16, 2015. Ex. 1001, 1. Following a preliminary amendment, the claims submitted with the ’514 Application were amended once during original prosecution. Ex. 1002, 43-47, 295-298[5]. The examiner issued one office action before allowing the application, rejecting the claims under 35 U.S.C. §§ 112 and 103. *Id.*, 92-103. The examiner rejected the claims as obvious over U.S. Patent 6,950,660 to Hsu et al. (“Hsu,” Ex. 1013) in view of U.S. Patent Application Publication No. 2009/0049539 to Halbedel et al. (“Halbedel,” Ex. 1014).

As to the independent claims, the examiner found that Hsu did not disclose “discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,” but that Halbedel did. Ex. 1002, 98. The examiner cited paragraph [0015] of Halbedel as disclosing this element. *Id.* In that passage, Halbedel explains that a hub maintains an access control list storing (1) a set of approved usernames and passwords authorized to access a particular server, or alternatively, (2) a set of valid IP addresses from which the server may be accessed. Ex. 1014, ¶ [0015].

In response, the applicant amended the independent claims to add to the “discarding” step the final “wherein” clause regarding the preset list. Ex. 1002, 42-47. For reference, the amendment to independent claim 1 is reproduced below:

> 1. (Currently Amended) A packet receiving method, comprising:
>
> receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;
>
> resolving the received server domain name to obtain a service server Internet

[5] For ease of reference, Requester cites the PDF page number of Exhibit 1002.

> protocol (IP) address; and
>
> discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device.

Ex. 1002, 43. The applicant argued that the claimed preset list differed from Halbedel's control list because it stored different information:

> ***Halbedel*** merely discloses a control list *that stores usernames, passwords, and user's IP addresses, in order to determine whether to grant the user access [to] the particular application server*.
>
> By contrast, as defined in amended claim 1, the preset list provides *the terminal domain name of each terminal device and a plurality of corresponding accessible service server IP addresses under an access authority of the terminal device*, so that after receiving a service request packet from the terminal device, resolving the domain name of a server carried in the service request packet, and obtaining IP addresses of the server, it can be determined whether the server is under access authority of the terminal device, by determining whether the IP address of the server resolved is in the preset list corresponding to the terminal's domain name.

Id. (emphasis in original)[6]. Following these amendments and arguments, the examiner allowed the application, citing the "discarding" step with the added "wherein" clause in the examiner's statement of reasons for allowance. *Id.*, 24-31.

As the prior art shows and Requester's technical expert explains, using a preset list, containing the terminal domain name of each terminal device and a plurality of corresponding authorized service server IP addresses, to control terminal device access to server IP addresses was well known years before the time of the '040 Patent. Ex. 1003, ¶ 45. For example, as explained below, in the SOCKS protocol (Exs. 1005-1007) developed in the mid-1990s, a SOCKS proxy server used a Configuration List mapping SOCKS client terminal domain names to corresponding accessible service IP addresses in determining whether to allow or deny incoming connections requests from terminal devices. *Id.* Additionally, Treuhaft (Ex. 1008) discloses a DNS name server that maintains subscriber information for each user or subscriber of the system identifying

[6] In this Request, emphasis is added unless otherwise specified.

corresponding authorized/unauthorized server IP addresses that the user or subscriber is authorized/unauthorized to access, and responds to DNS queries from those users and subscribers accordingly based on the subscriber information. *Id.*

4. Level of Ordinary Skill in the Art

As explained by Requester's technical expert a person of ordinary skill in the art ("POSA") of the'040 Patent in June 2012 would have had a bachelor's degree in computer science or computer engineering and two years of experience in computer and network security. In my opinion, however, less work experience may be compensated by a higher level of education, such as a master's degree, and vice versa.

B. Claim Construction

Because the '040 Patent has not yet expired, the Challenged Claims should be given their "broadest reasonable interpretation consistent with the specification." 37 CFR §1.75(d)(1); *see also* MPEP § 2111; *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316, 75 USPQ2d 1321, 1329 (Fed. Cir. 2005). Under this standard, "[t]he Patent and Trademark Office … determines the scope of claims in patent applications not solely on the basis of the claim language, but upon giving claims their broadest reasonable construction in light of the specification as it would be interpreted by one of ordinary skill in the art." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316, 75 USPQ2d 1321, 1329 (Fed. Cir. 2005) (quotation and citation omitted). "Because applicant has the opportunity to amend the claims during prosecution, giving a claim its broadest reasonable interpretation will reduce the possibility that the claim, once issued, will be interpreted more broadly than is justified." MPEP § 2111 (citing *In re Yamamoto*, 740 F.2d 1569, 1571 (Fed. Cir. 1984)).

With the exception of the terms identified below, Requester does not believe that the Challenged Claims require express construction. Requester presents the following claim analysis, as well as an application of the Challenged Claims to the prior art, in a manner consistent with the broadest reasonable interpretation standard. Requester reserves the right to advocate a different claim interpretation in another forum, if necessary.

1. "discarding the service request packet" (independent claims 1, 6, and 11)

Under the broadest reasonable interpretation standard, this phrase should be understood to include preventing unauthorized access to the resolved service server IP address. Ex. 1003, ¶¶ 46-47. A POSA would understand the claim language itself to support this understanding by reciting that the service request packet received from the terminal device—and containing the server domain name from which the IP address was resolved—is discarded if the resolved IP address does not belong to (i.e., is not) a preset address in the preset list. *See* Ex. 1001, 10:43-48. In other words, because the preset list does not list the resolved IP address referenced in the service request packet as a preset (e.g., authorized) IP address, the service request packet is discarded and the requested connection is not granted, preventing the terminal device from accessing the resolved IP address. Ex 1003, ¶ 46.

Additionally, dependent claims 4 and 9 recite establishing the connection if the resolved service server IP address belongs to (i.e., is) a preset address on the list. Conversely, this suggests that the connection is <u>not established </u>if the resolved IP address is <u>not</u> a preset address on the list (i.e., the DPI device prevents access). Ex. 1003, ¶ 47. It follows that, preventing access when the resolved IP address is not on the list falls within the scope of independent claim 1, from which claims 4 and 9 depend. The specification of the ’040 Patent also supports this understanding, explaining that if the resolved IP address is not on the preset list, “then the packet is considered to be abnormal, and the abnormal packet is discarded so as <u>to prevent the terminal device A from successfully accessing</u> the charged service through altering the packet without authorization[.]” Ex. 1001, 5:10-17. In other words, the service request is determined to be abnormal and discarded, preventing the terminal device from accessing the IP address, if the resolved IP address is not on the list. Ex. 1003, ¶ 47.

2. “if the resolved service server IP address does not belong to a preset service server IP address … in a preset list” (independent claims 1, 6, and 11)

“if the resolved service server IP address belongs to the preset service server IP address … in the preset list” (dependent claims 4 and 5)

To the extent these phrases are not indefinite, Requester believes they should be understood to mean “if the resolved service server IP address <u>is not</u> a preset service server IP address … in a preset list” and “if the resolved service server IP address <u>is</u> a preset service server IP address … in the preset list,” respectively. Ex. 1003, ¶48. Perhaps stemming from translation of its original Chinese text, the ’040 patent uses fairly odd language to describe the scenarios in which the preset list does or does not list the resolved IP address as a preset IP address for a terminal domain name. Rather than state that the IP address is or is not a preset address on the list, the ’040 patent respectively states that the resolved IP "belongs" or "does not belong" to a preset address on the list. *See, e.g.*, Ex. 1001, Abstract, 2:1-4, 2:15-19, 2:36-38, 4:16-17, 5:5-9, 5:32-35, 6:42-53, 6:63-66, 7:64-67, 8:34-38, 8:47-52, 8:55-59, 9:35-38, 10:12-15. From the context, however, a POSA would have understood that the '040 patent means the resolved IP address is not a preset address on the preset list when stating the resolved IP address "does not belong to a preset service server IP address … in a preset list," as recited in independent claims 1, 6, and 11. Ex. 1003, ¶ 48. And, by the same token, a POSA would have understood the '040 Patent means the resolved IP address is a preset address on the preset list when stating the resolved IP address "belongs to the preset server IP address … in the preset list." *Id.*

C. Listing of Prior Art Patents and Printed Publications

Requester requests reexamination of the Challenged Claims view of the following references. In support of the public availability of the non-patent literature prior art references in Exhibits 1005 (RFC 1928), 1006 (Koblas), 1007 (RFC 3089), and 1011 (Subramanian) before the claimed priority date of the '040 Patent, Requester submits a Declaration of Ingrid Hsieh-Yee, Ph.D., Professor of Library and Information Science at the Catholic University of America, as Exhibit 1015. This Request is also supported by a Declaration of Requester's technical expert, Angelos Keromytis, Ph.D., attached as Exhibit 1003.

- **Ex. 1005, Leech et al., "SOCKS Protocol Version 5," RFC 1928, March 1996 ("RFC 1928")**

RFC 1928 qualifies as prior art under pre-AIA 35 U.S.C. §§ 102(a) and (b).[7] Ex. 1015, ¶¶ 28-35. RFC 1928 was published in 1996 in the RFC (Request for Comment) series. Ex. 1005, 1; Ex. 1015, ¶ 35. Originally established in 1968, the RFC series edits and publishes technical and organizational documents about the Internet, including the specifications and policy documents

[7] The '514 Application is a transition application filed on December 1, 2014 (AIA) as a continuation of PCT/CN2012/077994 filed June 30, 2012 (pre-AIA). Ex. 1001, 1; *see* MPEP § 2159. The applicant did not file any statement under 37 CFR § 1.55 that the '514 Application contains or claims any subject matter with an effective filing date on or after March 16, 2013. Thus, provisions of pre-AIA 35 U.S.C. §§ 102 and 103 presumably apply to this Request. *See* MPEP § 2159. Regardless of whether pre-AIA or AIA provisions apply to this proceeding, however, the references in this Request qualify as prior art to the '040 Patent.

produced by the Internet Engineering Task Force (IETF), the Internet Research Task Force (IRTF), the Internet Architecture Board (IAB), and Independent Submissions. Ex. 1001, 1; Ex. 1015, ¶ 28; Ex. 1003, ¶ 50. Since long before the time of the '040 Patent, POSAs of network security were aware of the RFC series and consulted its publications in the ordinary course of their work. Ex. 1003, ¶ 50; Ex 1015, ¶ 26.

RFC 1928 was also archived by the Internet Archive hundreds of times since at least September 2000. Ex. 1015, ¶¶ 32-33; *see also* Ex. 1016; MPEP § 2128.II.E ("Prior art obtained via the Wayback Machine® sets forth a *prima facie* case that the art was publicly accessible at the date and time provided in the time stamp."). Additionally, at least ten other documents citing RFC 1928 published in 1996 and 1997, demonstrating that POSAs did in fact receive RFC 1928 and understand the information in it before the time of the '040 Patent. Ex. 1015, ¶ 34. Even though "there is no need to prove that someone actually looked at a publication" for it to be considered publicly available, MPEP § 2128.III (citations omitted), the citations to RFC 1928 suggest that POSAs did far more than merely look at it.

- **Ex. 1006, Koblas et al., "SOCKS," 1992, UNIX Security Symposium III Proceedings, 1992 ("Koblas")**

Published in 1992, Koblas also qualifies as prior art under §§ 102(a) and (b). Ex. 1015, ¶ 36-51. Koblas was presented at the USENIX UNIX Security Symposium III conference, sponsored by the USENIX Association in cooperation with the Computer Emergency Response Team (CERT), on September 14-16, 1992 in Baltimore, MD. Ex. 1015 ¶ 51. Koblas was presented during a "TCP/IP Network Security" session on Tuesday, September 15, 1992. *Id.*, ¶¶ 37, 51. POSAs in network security are familiar with the USENIX Association and CERT, and would have attended this conference at which Koblas was presented. Ex. 1003, ¶ 51. Additionally, Koblas was received and cataloged by the British Library in November 1993. Ex. 1003, ¶ 51; Ex. 1015, ¶¶ 39-45, 51. Koblas was also archived by the Internet Archive before the claimed priority date of the '040 Patent. Ex. 1016; Ex. 1015, ¶¶ 46-49; *see also* MPEP § 2128.II.E. Koblas was also cited nearly 80 times, including in articles published before the claimed priority date of the'040 Patent. Ex. 1015, ¶¶ 50-51; *see also* MPEP § 2128.III.

- **Ex. 1007, Kitamura, "A SOCKS-based IPv6/IPv4 Gateway Mechanism," RFC 3089, April 2001 ("RFC 3089")**

Published in 2001, RFC 3089 qualifies as prior art under §§ 102(a) and (b). Ex. 1007, 1;

see also Ex. 1015, ¶¶ 52-59. Similar to RFC 1928 discussed above, RFC 3089 is a document published in the RFC series. *Id.*; Ex. 1015, ¶¶ 52-55. For similar reasons to those discussed above for RFC 1928, RFC 3089 was also publicly available before the claimed priority date of the '040 Patent. For example, POSAs were familiar with the RFC series and its publications, so RFC 3089 would have been available to interested artisans. Ex. 1003, ¶ 50 ; Ex. 1015, ¶ 28. Additionally, RFC 3089 was archived by the Internet Archive nearly 90 times in June 2001. Ex. 1015, ¶¶ 56-57; *see also* MPEP § 2128.II.E. Moreover, at least 114 other documents cite RFC 3089, including at least 24 publications from 2002 to 2004, suggesting that POSAs received and considered RFC 3089. Ex. 1015, ¶ 58; *see also* MPEP § 2128.III (Although "there is no need to prove that someone actually looked at a publication" for it to be considered publicly available, Requester has made such a showing here) (citations omitted).

- **Ex. 1008, U.S. Patent Application Publication No. 2009/0157889 ("Treuhaft")**

Treuhaft is a U.S. patent application filed in 2008 and published in 2009. Ex. 1008, 1. Thus, Treuhaft qualifies as prior art under §§ 102(a), (b), and (e).

- **Ex. 1009, U.S. Patent Application Publication No. 2006/0167871 ("Sorenson")**

Sorenson is a U.S. patent application filed in 2004 and published in 2006. Ex. 1009, 1. Thus, Sorenson qualifies as prior art under §§ 102(a), (b), and (e).

- **Ex. 1010, U.S. Patent Application Publication No. 2004/0006621 ("Bellinson")**

Bellinson is a U.S. patent application filed in 2002 and published in 2004. Ex. 1010, 1. Thus, Bellinson qualifies as prior art under §§ 102(a), (b), and (e).

- **Ex. 1011, Subramanian et al., "An empirical vulnerability remediation model, IEEE International Conference on Wireless Communications, Networking and Information Security, 2010 ("Subramanian")**

Published in 2010, Subramanian qualifies as prior art under §§ 102(a) and (b). Ex. 1007, 1; *see also* Ex. 1015, ¶¶ 60-77. Subramanian was presented at the "2010 IEEE International Conference on Wireless Communications, Networking and Information Security," a conference held by Institute of Electrical and Electronics Engineers (IEEE) in June 2010 in Beijing, China. Ex. 1015, ¶¶ 66-67. Following the conference, Subramanian was uploaded to the IEEE Xplore digital library on August 5, 2010. *Id. Long* before the '040 Patent, POSAs were familiar with IEEE, attended IEEE conferences, and consulted IEEE publications in the ordinary course of their work. Ex. 1003, ¶ 56. Thus, POSAs would have attended the conference at which Subramanian was presented. *Id.* Additionally, if interested in the subject matter or author, they could have obtained Subramanian in Xplore using keyword searches or otherwise. *Id.*; *see also* Ex. 1015, ¶¶ 64-65 (Subramanian was indexed in Xplore by authors, abstract, and certain keywords). Indeed, interested parties have accessed Subramanian in Xplore at least 73 times since 2011. Ex. 1015 ¶¶ 64-65; *see also* MPEP § 2128.III (Although "there is no need to prove that someone actually looked at a publication" for it to be considered publicly available, Requester has made such a showing here) (citations omitted). Additionally, Subramanian was also cataloged by several libraries, including Stanford University Libraries in March 2011 and University of Alberta Libraries in October 2010. Ex. 1015 ¶¶ 66-77.

None of the references listed above was cited during prosecution of the '040 Patent. A Form SB-08 and copies of the cited references are submitted herewith. To the extent that additional references are discussed in the Keromytis Declaration, copies of these additional references are also being submitted, and are included on Form SB-08.

D. Ground 1: RFC 1928 in View of Koblas and RFC 3089 (the "SOCKS References") Presents a Substantial New Question of Patentability

RFC 1928, Koblas, and RFC 3089 all relate to the SOCKS protocol—an Internet protocol for exchanging network packets over TCP/IP between a client and server through a proxy server, called a SOCKS proxy server. *See* Exs. 1005, 1006, 1007. As they all refer to the same protocol, a POSA would have considered these together when considering whether and how to use a SOCKS system. Accordingly, Requester sometimes refers to RFC 1928, Koblas, and RFC 3089 collectively as "the SOCKS references."

As explained below, Requester submits that the SOCKS references raise a substantial new question of patentability as to claims 1, 4-6, and 9-11 because "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* MPEP 2242. For example, as discussed below here and in the Detailed Application section, the SOCKS references, when considered as an ordered combination, teach each limitation of the claims, including the purportedly novel feature that led the examiner to allow the claims: "discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device."

Further none of the SOCKS references was cited or previously considered during prosecution. Thus, the "same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent" at least because none of the art referenced herein was before the Office during prosecution of the '040 Patent. And, to Requester's knowledge, the '040 Patent has not been challenged in a prior post-grant proceeding.

1. Overview of the SOCKS References

a. RFC 1928

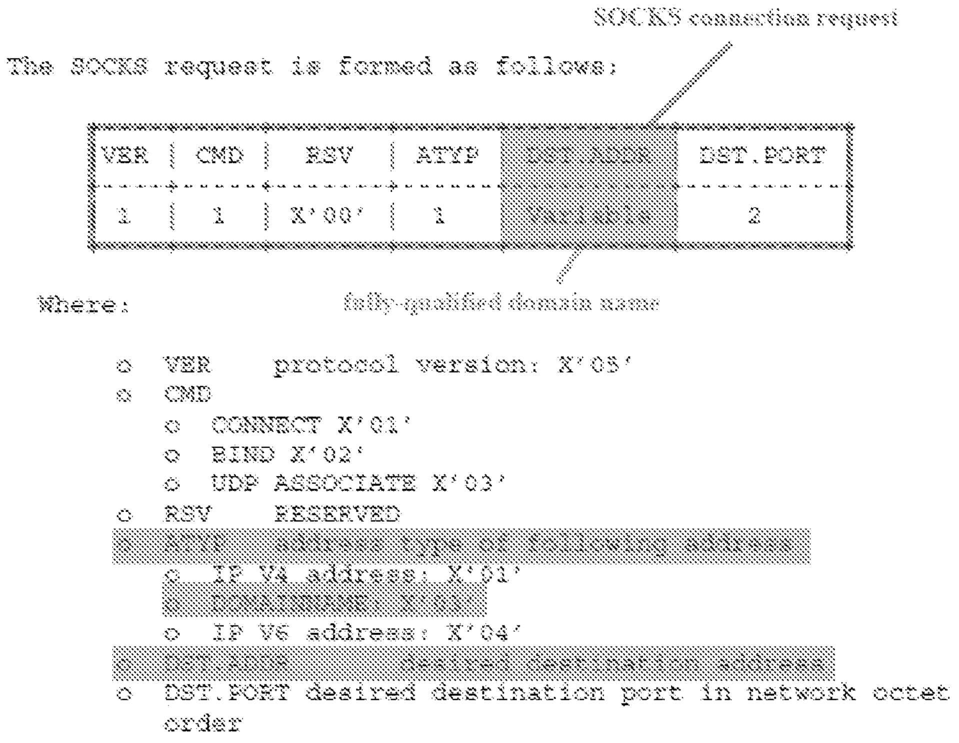
SOCKS is an Internet protocol that exchanges network packets over TCP/IP between a client and server through a proxy server, called a SOCKS proxy server. Ex. 1003, ¶ 62. RFC 1928 describes SOCKS Protocol Version 5 and aims to "provide a general framework … to transparently and securely traverse a firewall" in SOCKS[8]. Ex. 1005, 1; *see also id.*, 2 ("The [SOCKS] protocol described here is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall.").

When a SOCKS client wishes to connect to a server or other object behind the SOCKS proxy server, it sends a connection request to the SOCKS proxy server:

> When a TCP-based client wishes to establish a connection to an object that is reachable only via a firewall (such determination is left up to the implementation), it must open a TCP connection to the appropriate SOCKS port on the SOCKS server system. The SOCKS service is conventionally located on TCP port 1080. If the connection request succeeds, the client enters a negotiation for the authentication method to be used, authenticates with the chosen method, then sends a relay request. The SOCKS server evaluates the request, and either establishes the appropriate connection or denies it.

Id., 2-3; *see also id.*, 4 ("Once the method-dependent subnegotiation has completed, the client sends the request details."). The SOCKS connection request has the following form:

[8] In SOCKS, the SOCKS proxy server is sometimes called a "firewall". Ex. 1003, ¶ 60, n. 5. Requester refers to this SOCKS firewall component as the "SOCKS server" or "SOCKS proxy server," and uses these terms interchangeably throughout the Request.


SOCKS connection request
fully-qualified domain name

```
The SOCKS request is formed as follows:

+----+-----+-------+------+----------+----------+
|VER | CMD |  RSV  | ATYP | DST.ADDR | DST.PORT |
+----+-----+-------+------+----------+----------+
| 1  |  1  | X'00' |  1   | Variable |    2     |
+----+-----+-------+------+----------+----------+

Where:

   o  VER    protocol version: X'05'
   o  CMD
      o  CONNECT X'01'
      o  BIND X'02'
      o  UDP ASSOCIATE X'03'
   o  RSV    RESERVED
   o  ATYP   address type of following address
      o  IP V4 address: X'01'
      o  DOMAINNAME: X'03'
      o  IP V6 address: X'04'
   o  DST.ADDR       desired destination address
   o  DST.PORT desired destination port in network octet
      order
```

Id., 4*. As highlighted in the Figure above, the SOCKS request has a DST.ADDR field, which contains the "desired destination address" sought by the SOCKS client. *Id.*; Ex. 1003, ¶ 61. The preceding ATYP field specifies the "address type of [the] following address" contained in the DST.ADDR field, Ex. 1005, 4, which can take the form of "a fully-qualified domain name," Ex. 1005, 5. Ex. 1003, ¶ 61. Additionally, the SOCKS requests contains a source IP address of the SOCKS client because the connection request is sent over TCP/IP. Ex. 1003, ¶ 61.

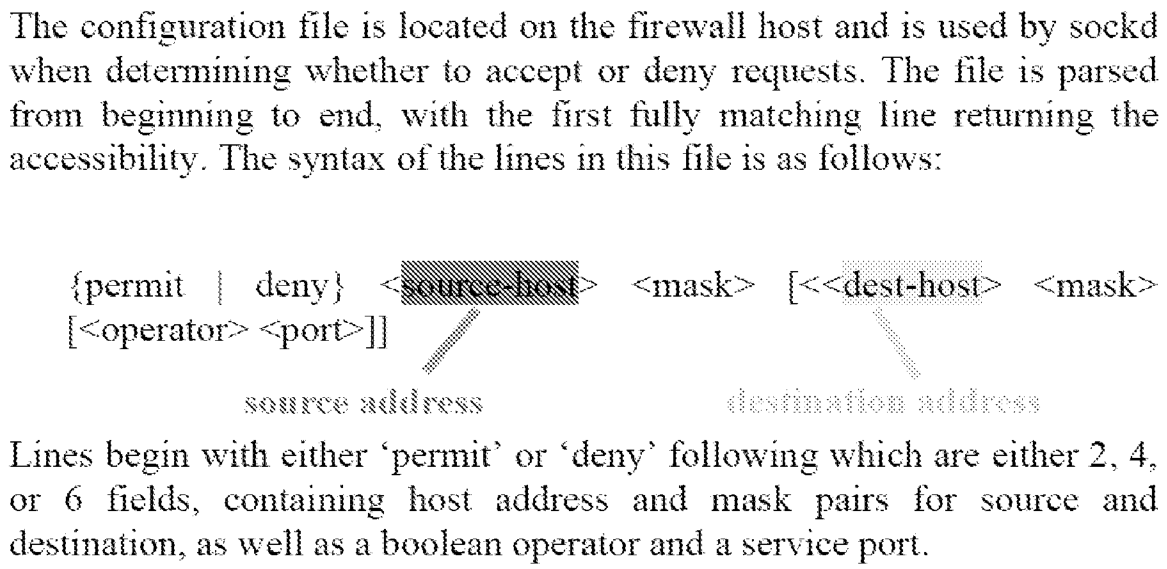
RFC 1928 teaches that "The SOCKS server evaluates the request, and either establishes the appropriate connection or denies it." Ex. 1005, 3. But because RFC 1928 focuses on laying out a general framework and protocol format for interacting with a SOCKS proxy server, RFC 1928 itself does not expressly describe the mechanism for evaluating connection requests. Ex. 1003, ¶ 62.

b. Koblas

Koblas uses SOCKS to address "[o]ne of the more important [security] issues" when connecting to a network over the Internet: "intruders attempting to gain access to local hosts." Ex.

1006, 3[9]. Koblas proposes "several strategies which can be used to configure an Internet connection to prevent unwanted intrusion" using the SOCKS protocol. Ex. 1006, 3.

In one strategy, Koblas teaches that the SOCKS server uses a "Configuration File" to allow or deny the connection request. The Configuration File contains an entry for each SOCKS client source identifying corresponding "permit" or "deny" destination addresses to which the SOCKS server will respectively permit or deny connections requested by the SOCKS client source:



> The configuration file is located on the firewall host and is used by sockd when determining whether to accept or deny requests. The file is parsed from beginning to end, with the first fully matching line returning the accessibility. The syntax of the lines in this file is as follows:
>
> {permit | deny} <source-host> <mask> [<<dest-host> <mask> [<operator> <port>]]
>
> Lines begin with either 'permit' or 'deny' following which are either 2, 4, or 6 fields, containing host address and mask pairs for source and destination, as well as a boolean operator and a service port.

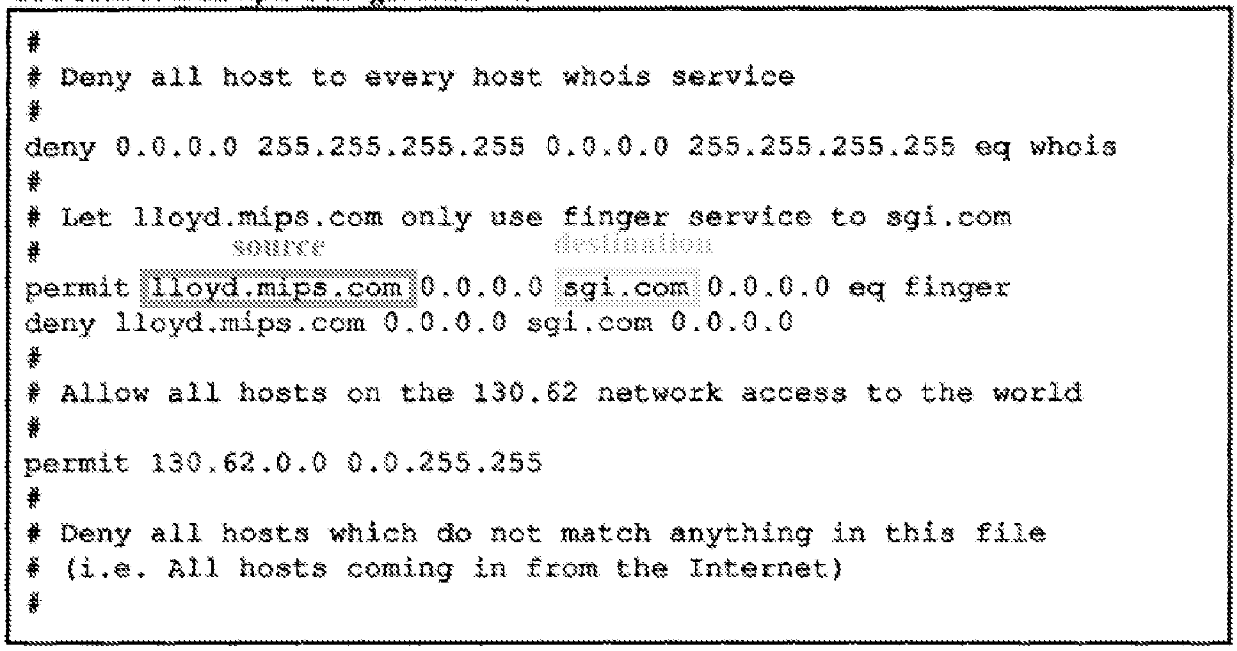
Ex. 1006, 7*.

As highlighted above, in the SOCKS Configuration File, the <source-host> field contains the host address of the SOCKS client source (claimed terminal device) and the corresponding <dest-host> field in the entry contains the address of the corresponding destination server (claimed corresponding preset service server IP address). *Id.*; Ex. 1003, ¶¶ 64-65. Depending on which value the {permit | deny} field contains, the SOCKS server will respectively permit or deny the SOCKS client source named in the <source-host> field to connect to the corresponding destination server address in the <dest-host> field. Ex. 1006, 7; Ex. 1003, ¶ 65. Additionally, Koblas explains that "[h]ost addresses and services may be specified either by <u>name or number</u>," meaning SOCKS supports listing the addresses in the <source-host> and <dest-host> fields as a domain name or an IP address. Ex. 1006, 8; Ex. 1003, ¶ 65.

[9] For ease of reference, Requester cites the PDF page number of Exhibit 1006.

In Figure 5, Koblas "shows an example of how the lines in a configuration file might appear":

FIGURE 5. A Sample Configuration File


source
destination

```
#
# Deny all host to every host whois service
#
deny 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255 eq whois
#
# Let lloyd.mips.com only use finger service to sgi.com
#
permit lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0 eq finger
deny lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0
#
# Allow all hosts on the 130.62 network access to the world
#
permit 130.62.0.0 0.0.255.255
#
# Deny all hosts which do not match anything in this file
# (i.e. All hosts coming in from the Internet)
#
```

Ex. 1006, 8. In this Sample Configuration file, the SOCKS server permits a request from the source device with the domain name lloyd.mips.com to connect to the corresponding destination sgi.com, and denies requested connections "which do not match anything in this file." *Id.*; Ex. 1003, ¶ 66. Although shown as a domain name in this example, the server destination address sgi.com "may be specified either by name or number," so the IP address of sgi.com could be used instead. Ex. 1006, 8; Ex. 1003, ¶ 66.

c. RFC 3089

RFC 3089 explains that, "[i]n all communication applications, it is [] necessary to obtain destination IP address information to start a communication." Ex. 1007, 4. To that end RFC 3089 describes a process by which the SOCKS server uses DNS to resolve the fully-qualified domain name (FQDN) of the destination node (Destination D) into its "real IP address" when receiving a connection request from the source node (Client C):

> The detailed internal procedure of the "DNS name resolving delegation" and address mapping management related issues are described as follows.
>
> 1. An application on the source node (Client C) tries to get the IP address information of the destination node (Destination D) by calling the DNS name resolving function (e.g.,

gethostbyname()). At this time, the logical host name ("FQDN") information of the Destination D is passed to the application's *Socks Lib* as an argument of called APIs.

2. Since the *Socks Lib* has replaced such DNS name resolving APIs, the real DNS name resolving APIs is not called here. The argued "FQDN" information is merely registered into a mapping table in *Socks Lib*, and a "fake IP" address is selected as information that is replied to the application from a reserved special IP address space that is never used in real communications (e.g., 0.0.0.x). The address family type of the "fake IP" address must be suitable for requests called by the applications. Namely, it must belong to the same address family of the Client C, even if the address family of the Destination D is different from it. After the selected "fake IP" address is registered into the mapping table as a pair with the "FQDN", it is replied to the application.
3. The application receives the "fake IP" address, and prepares a "socket". The "fake IP" address information is used as an element of the "socket". The application calls socket APIs (e.g., connect()) to start a communication. The "socket" is used as an argument of the APIs.
4. Since the *Socks Lib* has replaced such socket APIs, the real socket function is not called. The IP address information of the argued socket is checked. If the address belongs to the special address space for the fake address, the matched registered "FQDN" information of the "fake IP" address is obtained from the mapping table.
5. The "FQDN" information is transferred to the *Gateway* on the relay server (Gateway G) by using the SOCKS command that is matched to the called socket APIs. (e.g., for connect(), the CONNECT command is used.)
6. Finally, the real DNS name resolving API (e.g., getaddrinfo()) is called at the *Gateway*. At this time, the received "FQDN" information via the SOCKS protocol is used as an argument of the called APIs.
7. The *Gateway* obtains the "real IP" address from a DNS server, and creates a "socket". The "real IP" address information is used as an element of the "socket".
8. The *Gateway* calls socket APIs (e.g., connect()) to communicate with the Destination D. The "socket" is used as an argument of the APIs.

Ex. 1007, 5-6.

2. The Combination of RFC 1928, Koblas, and RFC 3089 Presents a Substantial New Question of Patentability

The combination of the SOCKS references presents a substantial new question of patentability with respect to the Challenged Claims of the ’040 Patent. SOCKS, as described in RFC 1928, is similar to the claimed invention, in part, because it uses a SOCKS proxy server in a firewall-like role similar to the DPI device in the ’040 Patent. Ex. 1003, ¶ 68. Like the DPI device in the ’040 Patent, the SOCKS server receives a connection request from a source host device seeking access to a destination host server. *Id.* The SOCKS connection request, like the service request in the ’040 Patent, contains two pieces of information: (1) an identifier of the source device;

and (2) a domain name of the destination server. *Id.* Both the SOCKS server and the DPI device resolve the domain name of the destination server into its corresponding IP address. *Id.* And, like the DPI device, RFC 1928 teaches that the SOCKS server evaluates the connection request and denies it if the request is not appropriate. *Id.*

Although RFC 1928 does not expressly describe the mechanism to evaluate the connection request, Koblas discloses that the SOCKS server uses a Configuration File for this purpose. *Id.*, ¶ 69. The SOCKS Configuration File goes directly to the purported point of novelty of the '040 Patent and the reason the examiner allowed the Challenged Claims. Namely, like the "preset list" of the '040 Patent, the Configuration File of Koblas lists the domain name of each source device and corresponding accessible server IP addresses under an access authority of the source device. *Id.* When a connection request is received, the SOCKS server applies the Configuration File to determine whether to allow or deny the connection. *Id.* If the Configuration File lists the IP address of the destination server as an allowed address for the domain name of the source host making the connection request, the SOCKS server allows the connection. *Id.* If not, however, the SOCKS server denies the connection. *Id.* As explained below in the Detailed Application section for the specific claim elements pertinent to the combination, a POSA would have been motivated to combine the SOCKS references and had a reasonable expectation in doing so. *Id.*

Accordingly, as described above and below in the Detailed Application section, the SOCKS references disclose, or at least render obvious, "*discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device*," as recited by independent claims 1, 6, and 11. *Id.* Thus, the SOCKS references present a substantial new question of patentability with respect to the Challenged Claims.

E. Grounds 2-6: Treuhaft, Treuhaft in View of Sorenson, and Treuhaft/Sorenson in View of Bellinson Present Substantial New Questions of Patentability

As explained below, Requester submits that the Treuhaft, Treuhaft in view of Sorenson, and Treuhaft/Sorenson in view of Bellinson present substantial new questions of patentability as to claims 1, 4-6, and 9-11 because "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* MPEP 2242. For example, as discussed below here and in the Detailed Application section, Treuhaft, Treuhaft in view of Sorenson, and Treuhaft/Sorenson in view of Bellinson, when considered as an ordered combination, teach each limitation of the claims, including the purportedly novel feature that led the examiner to allow the claims: "discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device."

Further none of these references was cited or previously considered during prosecution. Thus, the "same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent" at least because none of the art referenced in was before the Office during prosecution of the '040 Patent or during any known prior post-grant proceeding challenging the '040 Patent.

1. Overview of the References

a. Treuhaft

Treuhaft describes a system in which a DNS name server 120 of Treuhaft maintains subscriber information 208 for various users or subscribers of the system. *See* Ex. 1008, ¶¶ [0028], [0029], [0034], [0036], [0039], [0054], [0060], [0064], FIG. 2 (subscriber information 280). "The subscriber information can include preferences or other settings for how a user or subscriber wishes to control domain name resolution within the DNS resolution features." *Id.*, ¶ [0060]. "For example, a user or subscriber may establish subscriber information that instructs DNS nameserver 120 to alter responses to DNS requests that are associated with adult web sites, potential phishing or pharming sites, and other sites deemed inappropriate by the user or containing material illegal in the country of the user." *Id.*, ¶ [0028]. Thus, the DNS name server 120 has a similar role to the SOCKS server and the DPI device of the '040 patent.

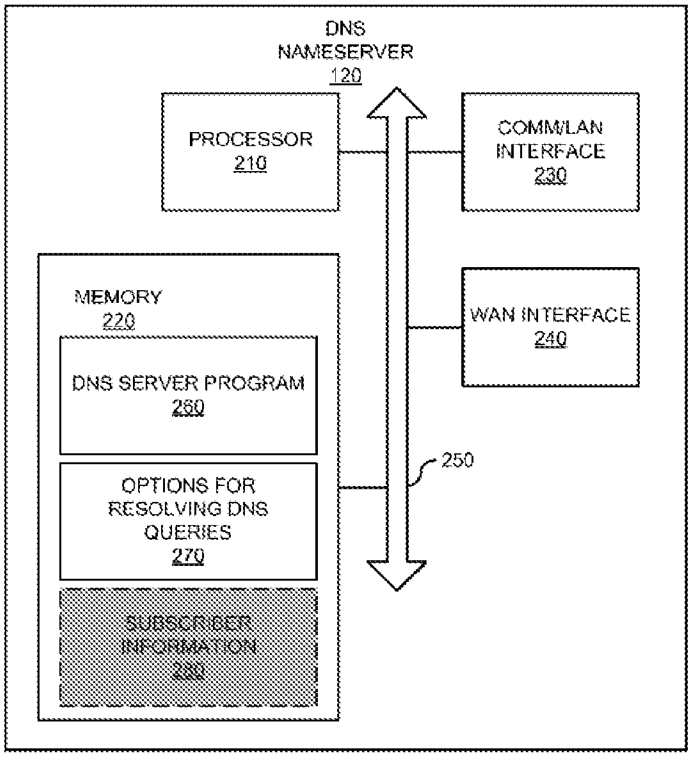
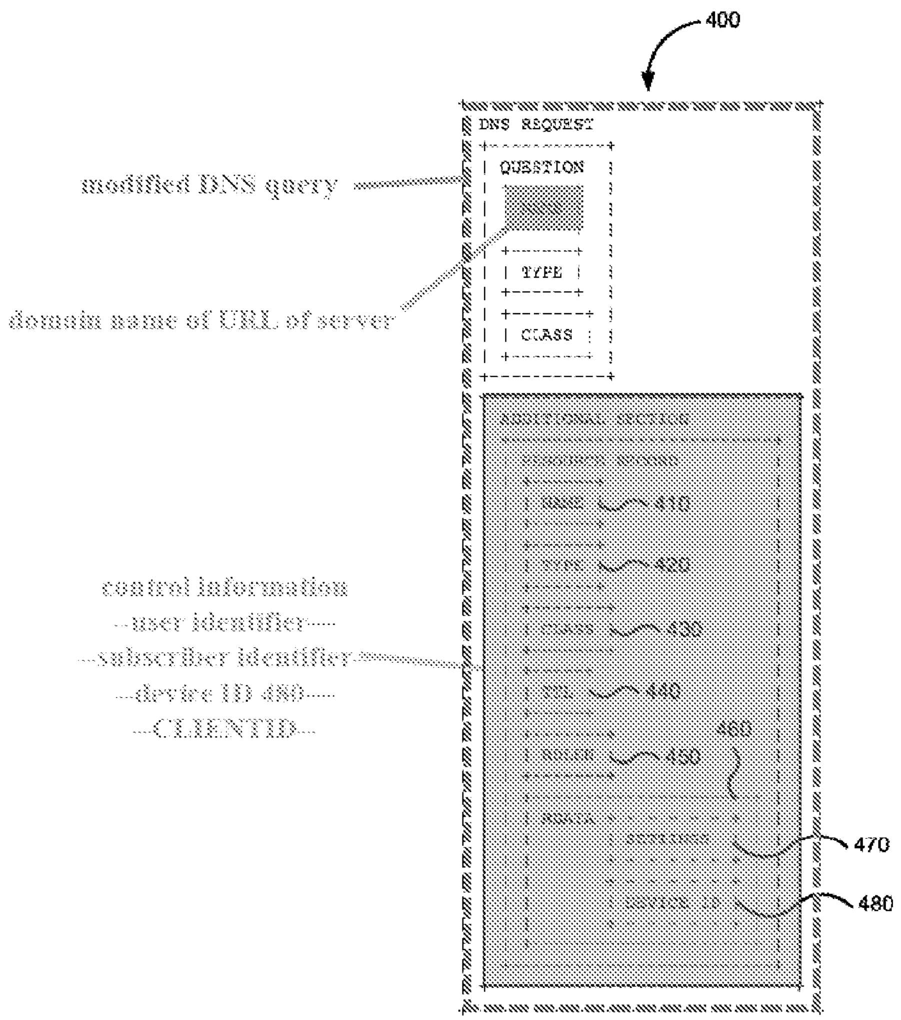


FIG. 2

Treuhaft, FIG. 2*

In Treuhaft, the DNS name server 120 receives a DNS query from a host device 105 seeking an IP address to connect to a server. *Id.*, ¶¶ [0063], [0064], FIG. 5A (step 525), FIG. 5B (step 530). The DNS query contains two pieces of information (1) a domain name for a URL of the server; and (2) "control information" identifying the host device 105:



Treuhaft, FIG. 4*

As highlighted in Figure 4 above, the DNS query 400 includes a NAME field containing the domain name of the URL the host device 105 seeks to access. *Id.*, ¶ [0054]. Additionally, before the host device 105 sends the DNS query to name server 102, "the DNS query is modified with control information." *Id.*, ¶ [0058], FIG. 5A (step 520); *see also id.*, ¶ [0067] ("control information may be encoded into an individual DNS query that enables a DNS nameserver to identify DNS resolution options, filters, or features to apply when resolving the individual DNS query"). "The control information may specify … a user or subscriber identifier, a device identifier, or the like." *Id.*, ¶ [0036].

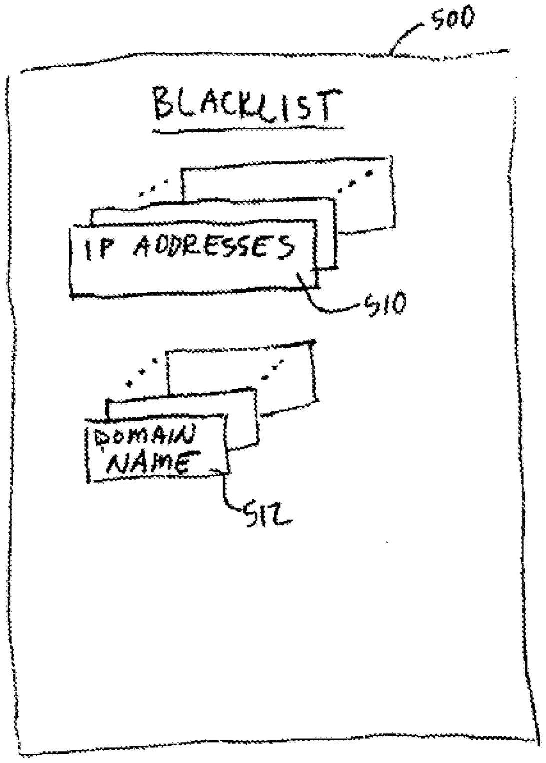
Upon receiving the DNS query, the "DNS nameserver 120 determines how to respond to host device 105" by applying the subscriber information 280 for the particular user or subscriber.

Id., ¶ [0032]; *see also id.*, ¶¶ [0028], [0029], [0034], [0036], [0039], [0054], [0060], [0064]. Specifically, the DNS name server 120 parses the control information in the DNS query to identify the particular user or subscriber associated with the DNS query, and then retrieves that user or subscriber's subscriber information 280. *Id.*, ¶ [0064], FIG. 5B (step 353); *see also id.*, ¶ [0060].

Using the user or subscriber's subscription information 280, the DNS name server 120 "make[s] a decision whether to use the corresponding IP address or another IP address when generating a DNS response based on applying one or more DNS resolution options or features". *Id.* ¶ [0065]. For example, rather than return the resolved IP address requested by the host device 105, "DNS nameserver 120 may determine to substitute the IP address of a website that provides information why the domain name is being block[ed], forwarded, filtered, or otherwise includes material the user has expressed a desire to control." Ex. 1008, ¶ [0065], FIG. 5B (steps 540). Then, the DNS name server 120 generates a DNS response "<u>substitut[ing]</u> [the] IP address based on applying one or more of the available DNS resolution options, filters, or features" and sends the DNS response with the substituted IP address to the host device 105. Ex. 1008, ¶ [0066], FIG. 5B (steps 545, 550); Ex. 1003, ¶ 73.

b. Overview of Sorenson

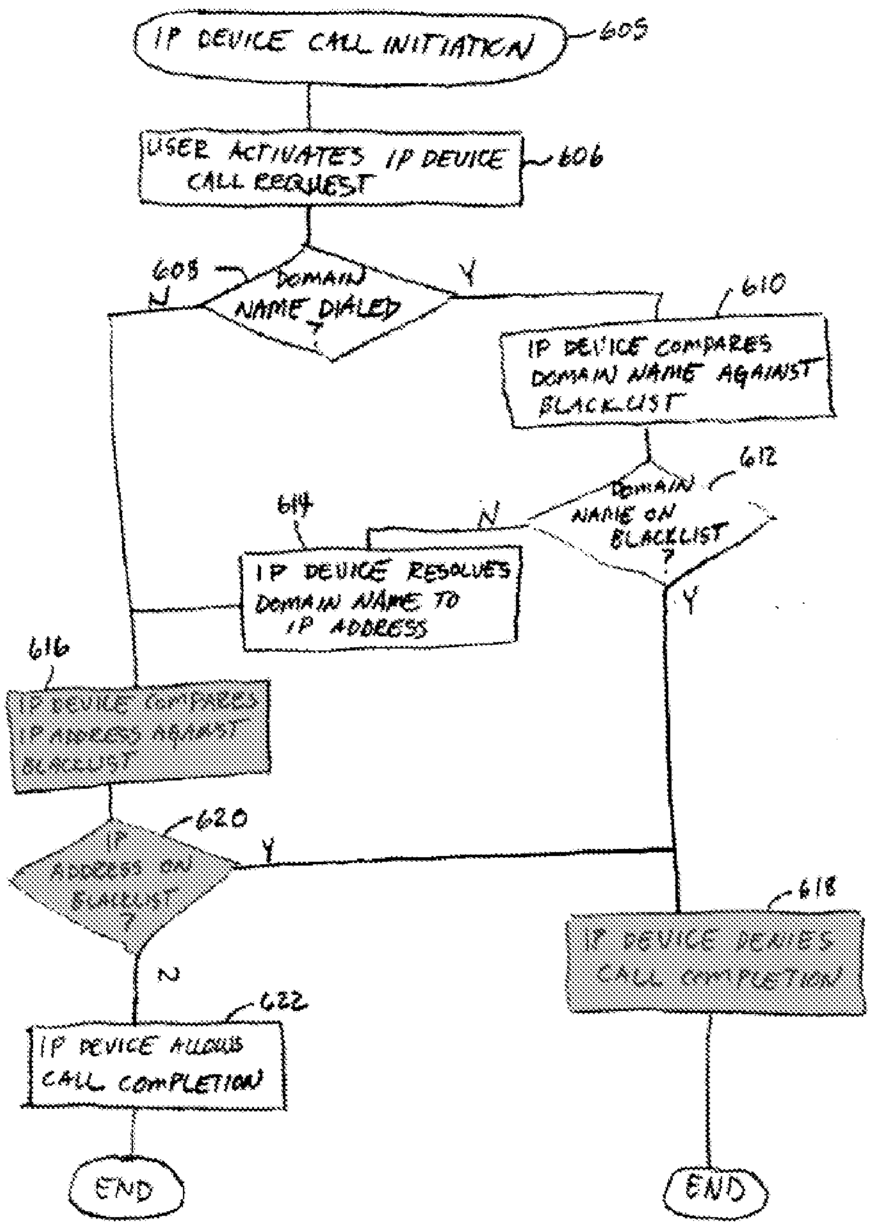
Sorenson discloses a "system and method for blocking access by a network device to specific network resources by comparing a specific resource identifier against entries in a blacklist and facilitating a connection accordingly." Ex. 1009, Abstract. In Sorenson, the system receives "a call request for the establishment of a communication session between IP device 12 and associated service 20. *Id.*, ¶ [0027]; *see also id.*, ¶¶ [0031], [0032]. The call request "include[s] a specific identifier such as an entered IP address, domain name, or conventional phone number or name resolved into one of an IP address or domain name." *Id.*, ¶ [0031].



Before granting the call request, the system of Sorenson performs a two-stage blacklist check to determine whether to establish the connection or discard the request. *See id.*, ¶¶ [0031]-[0032], FIG. 6. As shown in Figure 2 of Sorenson reproduced to above, the blacklist 500 contains both blacklisted domain name names 512 and blacklisted IP addresses 510. *Id.* 1009, ¶ [0028], FIG. 2.

First, as highlighted in step 610 of Figure 6 reproduced below, Sorenson performs a domain-name-blacklist check by "compar[ing] 610 the domain name against the blacklist 500′ (FIG. 2) to determine 612 if the domain name is located within the blacklist 500′." *Id.*, ¶ [0031]. "If the domain name utilized for initiating the call is located with the blacklist 500′, then the IP device denies 618 the completion of the call and may alternatively notify the user of such denial." *Id.* But "[i]f the domain name is not on the blacklist, then" Sorenson resolves the address and performs a second, <u>IP-address-blacklist</u> check before establishing the connection. *Id.*

Specifically, Sorenson "resolves … the domain name into an IP address for further comparison" in step 614, *id.*, and then "compares … the IP address against the blacklist 500'" in step 616, *id.*, ¶ [0032]. If the IP address is located within the blacklist 500', Sorenson denies the connection. *Id.*, ¶ [0032], FIG. 6 (step 618). If the IP address is not found on the blacklist 500', however, Sorenson establishes the connection. *Id.*, ¶ [0032], FIG. 6 (step 622).


IP DEVICE CALL INITIATION
605
USER ACTIVATES IP DEVICE CALL REQUEST
606
608
DOMAIN NAME DIALED ?
Y
N
610
IP DEVICE COMPARES DOMAIN NAME AGAINST BLACKLIST
612
DOMAIN NAME ON BLACKLIST ?
N
Y
614
IP DEVICE RESOLVES DOMAIN NAME TO IP ADDRESS
616
IP DEVICE COMPARES IP ADDRESS AGAINST BLACKLIST
620
IP ADDRESS ON BLACKLIST ?
Y
N
618
IP DEVICE DENIES CALL COMPLETION
622
IP DEVICE ALLOWS CALL COMPLETION
END
END

Sorenson, FIG. 6*

c. Overview of Bellinson

Similar to '040 Patent, SOCKS, Treuhaft, and Sorenson, Bellinson discloses "a system and method for controlling whether a user may access certain Internet sites" by applying "an allow-block list" to "determine[] whether the URL is referenced on the allow-block list and, if so, allow[] or disallow[] access to the site referenced by the URL accordingly." Ex. 1010, Abstract. In Bellinson, "[t]he allow-block list is a listing of specific site identifiers that the user is expressly authorized to view or prohibited from viewing." *Id.*, ¶ [0020]; *see also id.*, ¶¶ [0009] ("The allow-block list is a file containing a listing of specific URLs that the user is expressly authorized to view or expressly prohibited from viewing.").

As shown in Figure 3, reproduced to the right, in step 244 Bellinson's system receives an access request containing "a specified site identifier that references an Internet site. Examples of such site identifiers include designators such as www.microsoft.com but could also include an Internet Protocol (IP) address." *Id.*, ¶ [0049], FIG. 3. In step 246 of Figure 3, Bellinson "determines whether the site identifier is on the allow-block list at step 246. *Id.* "If the site identifier is referenced on the allow-block list," in step 248 Bellinson "determine[s] whether the site identifier is designated as blocked on the allow-block list." *Id.* "If the site identifier is [designated as] blocked," Bellinson blocks the connection. *Id.*, ¶ [0050]. Otherwise, Bellinson allows the connection. *Id.*, ¶ [0051].

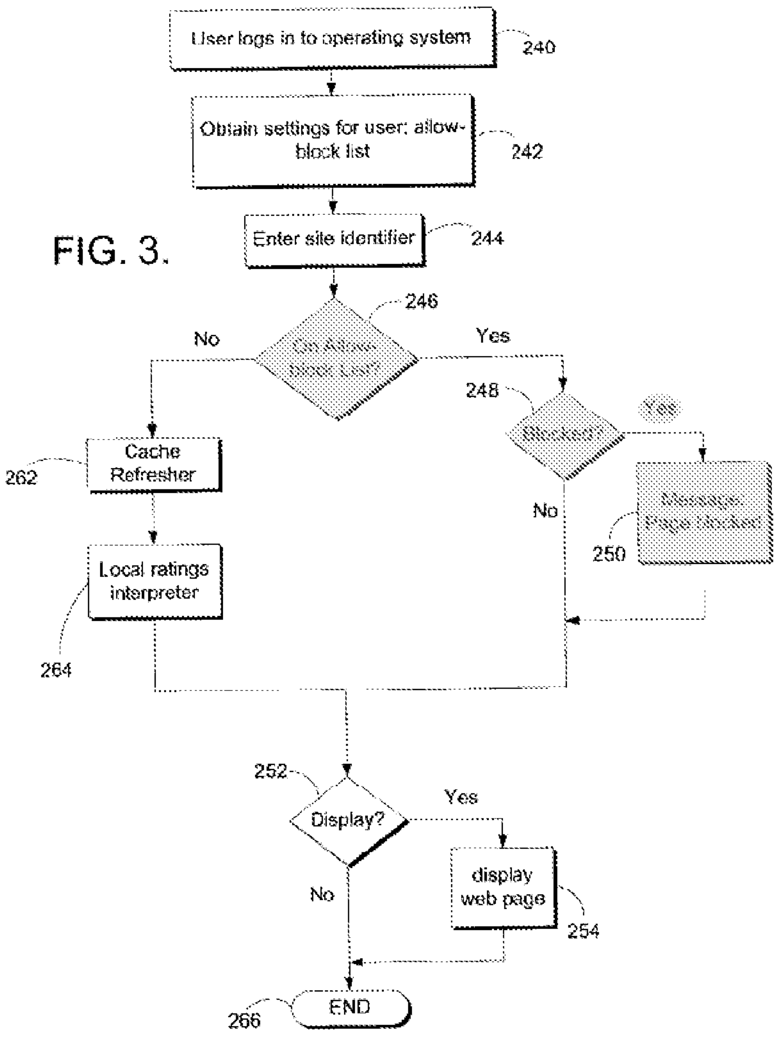


FIG. 3.

2. Treuhaft, Treuhaft in view of Sorenson, and Treuhaft/Sorenson in view of Bellinson Present Substantial New Questions of Patentability

Combinations of Treuhaft, Treuhaft and Sorenson, and Treuhaft/Sorenson in view of Bellinson raise substantial new questions of patentability with respect to the Challenged Claims of the '040 Patent. Treuhaft is similar to the system claimed in the '040 Patent, in part, because it uses a DNS name server 120 in a role similar to the DPI device in the '040 Patent. Ex. 1003, ¶ 82. Like the DPI device in the '040 Patent, Treuhaft's DNS name server 120 receives a request (a modified DNS query) from a host device seeking to access a destination server. *Id.* The modified

DNS query, like the service request in the ’040 Patent, contains two pieces of information: (1) control information identifying the host device; and (2) a domain name of the destination server the host device seeks to access. *Id.* Both Treuhaft’s DNS name server and the DPI device resolve the domain name of the destination server into its corresponding IP address. *Id.* And, similar to the DPI device, Treuhaft’s DNS name server evaluates the DNS query and denies it if the request is not appropriate. *Id.*

In fact, the manner in which Treuhaft’s DNS name server evaluates the DNS query goes directly to the purported point of novelty of the ’040 Patent and the reason the examiner allowed the ’040 Patent claims. *Id.*, ¶ 83. Like the DPI devices applies the “preset list” in the ’040 Patent, the DNS name server of Treuhaft applies subscriber information 280 to determine whether to allow or deny the requested connection. *Id.* And Treuhaft discloses or suggests that the subscriber information 280, like the “preset list” of the ’040 Patent, identifies corresponding server IP addresses under the access authority of each user or subscriber in Treuhaft. *Id.* Accordingly, Treuhaft alone presents a substantial new question of patentability with respect to the ’040 Patent.

Sorenson and Bellinson further address the purported point of novelty of the ’040 Patent and raise substantial new questions of patentability in combination with Treuhaft. For example, Sorenson’s blacklist of IP addresses and Bellinson’s allow-block list of addresses each correspond to the claimed preset list. *Id.*, ¶ 84. Though Sorenson’s IP address blacklist identifies corresponding addresses not under the access authority--rather than under the access authority--of the device seeking the connection, as evidenced by Bellinson (and Subramanian), whitelists and blacklists were well-known and used interchangeably long before the claimed priority date of the ’040 Patent. *Id.* Accordingly, instead of a blacklist, a POSA would have found it obvious to use a whitelist of IP addresses under the access authority of the host device in the Treuhaft combinations. *Id.* As explained above and in more detail below in the Detailed Application section for the specific claim elements pertinent to the combination, a POSA would have had been motivated to combine Treuhaft with Sorenson and/or combine Treuhaft/Sorenson with Bellinson, and had a reasonable expectation in doing so. *Id.*

Accordingly, as described above and below in the Detailed Application section, the proposed combinations of Treuhaft, Treuhaft and Sorenson, and Treuhaft/Sorenson in view of Bellinson teach, or at least render obvious, “*discarding the service request packet if the resolved* *service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list, wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device,*" as recited by independent claims 1, 6, and 11. *Id.* Thus, the proposed combinations of Treuhaft, Treuhaft and Sorenson, and Treuhaft/Sorenson in view of Bellinson present a substantial new question of patentability with respect to the Challenged Claims.

II. DETAILED APPLICATION OF THE PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

In Grounds 1 through 6 below, Requester explains in detail how the prior art references listed above establish the unpatentability of the Challenged Claims as anticipated under § 102 and obvious under § 103. Requester respectively requests the Office to issue an office action rejecting the Challenged Claims based on Grounds 1-6.

A. Ground 1: RFC 1928 in view of Koblas and RFC 3089 ("SOCKS") Render Obvious Claims 1, 4-6, and 9-11 of the ' 040 Patent Under § 103

RFC 1928 in view of Koblas and RFC 3089 ("SOCKS") render obvious claims 1, 4-6, and 9-11 of the '040 Patent under § 103.

1. Independent Claim 1

a. [1.pre] "*A packet receiving method, comprising:*"

To the extent the preamble is limiting, RFC 1928 describes SOCKS Protocol Version 5, which is a packet receiving method. Ex. 1003, ¶ 88. The SOCKS Protocol is an Internet protocol that exchanges (i.e., sends and receives) network packets over TCP/IP between a client and server through a proxy server that acts as a firewall, called a SOCKS proxy server. *Id.* SOCKS Version 5 provides authentication so only authorized users may access a server. *Id.* According to RFC 1928, SOCKS Version 5 is a "protocol … designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall." Ex. 1005, 2.

Accordingly, RFC 1928 discloses "*a packet receiving method,*" as claimed.

b. [1.1] "*receiving a service request packet sent by a terminal device,*

> ***wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;"***

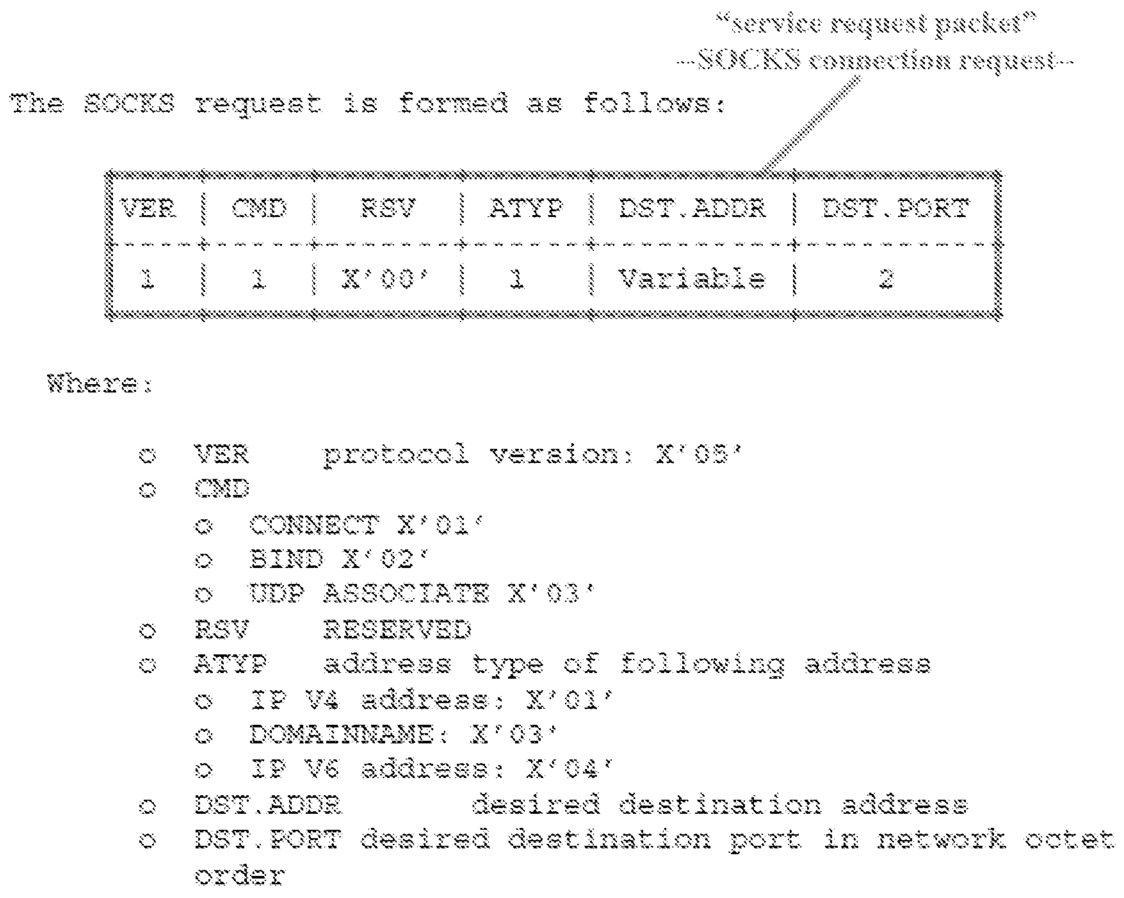
RFC 1928 and Koblas renders obvious this element. Ex. 1003, ¶¶ 90-95. As laid out below, RFC 1928 discloses that a SOCKS proxy server receives a SOCKS connection request (claimed service request packet) sent by a SOCKS client (claimed terminal device). *Id.*, ¶ 90. The SOCKS request carries a fully-qualified domain name (claimed server domain name) indicating a server behind the SOCKS proxy server to which the SOCKS client desires a connection (claimed service server required by the service request packet). *Id.* The SOCKS connection request also contains a source IP address of the SOCKS client (claimed terminal device)--rather than a terminal domain name--but it would have been obvious to include a domain name in the SOCKS connection request based on the disclosure of Koblas, as discussed below. *Id.*

Specifically, RFC 1928 discloses that a SOCKS proxy server receives a SOCKS connection request (claimed service request packet) from a TCP-based SOCKS client (claimed terminal device):

> When a <u>TCP-based client</u> wishes to <u>establish a connection</u> to an object that is reachable only via a firewall (such determination is left up to the implementation), it must open a TCP connection to the appropriate SOCKS port on the SOCKS server system. The SOCKS service is conventionally located on TCP port 1080. If <u>the connection request</u> succeeds, the client enters a negotiation for the authentication method to be used, authenticates with the chosen method, then sends a relay request. The SOCKS server evaluates the request, and either establishes the appropriate connection or denies it.

Ex. 1005, 2-3; *see also id.*, 4 ("Once the method-dependent subnegotiation has completed, <u>the client sends the request details</u>.").

RFC 1928 shows the form of the SOCKS connection request:


"service request packet"
--SOCKS connection request--

The SOCKS request is formed as follows:

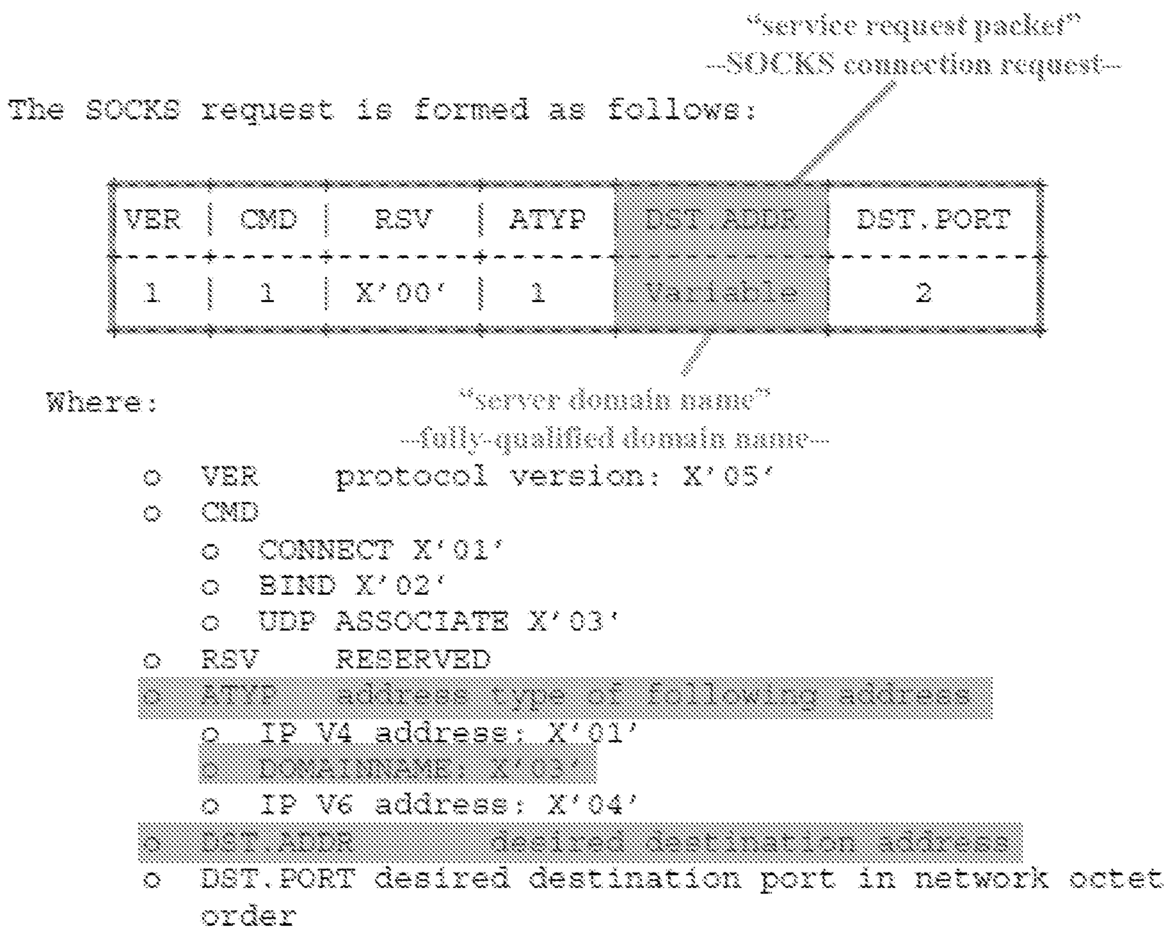
VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	X'00'	1	Variable	2

Where:

- o VER protocol version: X'05'
- o CMD
  - o CONNECT X'01'
  - o BIND X'02'
  - o UDP ASSOCIATE X'03'
- o RSV RESERVED
- o ATYP address type of following address
  - o IP V4 address: X'01'
  - o DOMAINNAME: X'03'
  - o IP V6 address: X'04'
- o DST.ADDR desired destination address
- o DST.PORT desired destination port in network octet order

Id., 4.*; *see also* Ex. 1003, ¶ 92. As highlighted above, the SOCKS connection request is a "packet" because it is a unit of information transmitted as a whole. Ex. 1012, 4 (definition of "packet"); Ex. 1003, ¶ 92. Specifically, the SOCKS connection request is a TCP/IP packet sent over the Internet containing the VER, CMD, RSV, ATYP, DST.ADDR, and DST.PORT fields in a single message, as shown in the Figure above. Ex. 1005, 4; *see also id.*, 3; Ex. 1003, ¶ 92 (the SOCKS client is a TCP/IP client and thus communicates using TCP/IP packets). Accordingly, the SOCKS proxy server's receiving a SOCKS connection request from a SOCKS client discloses "receiving a service request packet," as claimed. Ex. 1003, ¶ 92.

The SOCKS connection request "carries … a server domain name indicating a service server required by the service request packet sent by the terminal device," as claimed. *Id.* ¶ 93. Particularly, the SOCKS connection request carries a DST.ADDR (destination address) field containing a fully-qualified domain name (claimed server domain name) of a server, behind the SOCKS proxy server, to which the SOCKS client desires to connect:


"service request packet"
--SOCKS connection request--
"server domain name"
--fully-qualified domain name--

```
The SOCKS request is formed as follows:

        +----+-----+-------+------+----------+----------+
        |VER | CMD |  RSV  | ATYP | DST.ADDR | DST.PORT |
        +----+-----+-------+------+----------+----------+
        | 1  |  1  | X'00' |  1   | Variable |    2     |
        +----+-----+-------+------+----------+----------+

  Where:

     o  VER    protocol version: X'05'
     o  CMD
        o  CONNECT X'01'
        o  BIND X'02'
        o  UDP ASSOCIATE X'03'
     o  RSV    RESERVED
     o  ATYP   address type of following address
        o  IP V4 address: X'01'
        o  DOMAINNAME: X'03'
        o  IP V6 address: X'04'
     o  DST.ADDR       desired destination address
     o  DST.PORT desired destination port in network octet
        order
```

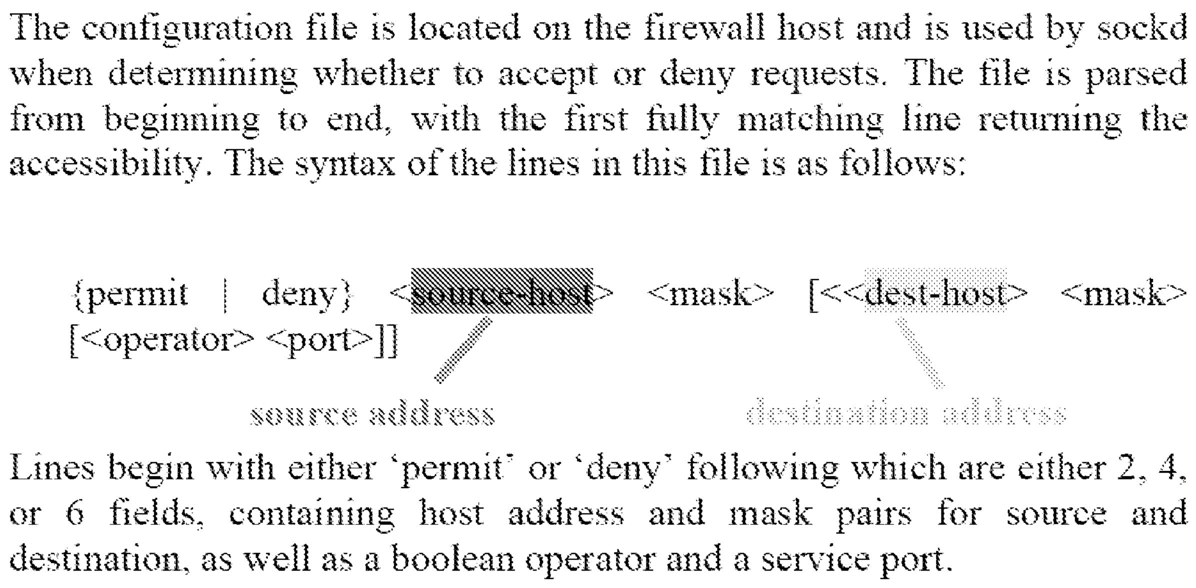
Ex. 1005, 4.

As highlighted in the Figure above, the SOCKS request has a DST.ADDR field, which contains the "desired destination address" sought by the SOCKS client. *Id.*; Ex. 1003, ¶ 94. The preceding field ATYP specifies the "address type of [the] following address" contained in the DST.ADDR field. Ex. 1005, 4; Ex. 1003, ¶ 94. As shown, SOCKS supports a destination address in the form of "DOMAINNAME" using protocol version "X' 03'". Ex. 1005, 4; Ex. 1003, ¶ 94. Following the description of the SOCKS connection request, RFC 1928 explains that X' 03' means "the address field contains a <u>fully-qualified domain name</u>." Ex. 1005, 5. Thus, in use, SOCKS clients send connection requests containing a fully-qualified domain name (e.g., www.website.com) in the DST.ADDR field. Ex. 1003, ¶ 94. Because the SOCKS connection request contains a fully-qualified domain name of the destination server, it discloses a "service request packet … carr[ying] … a server domain name indicating a service server required by the service request packet sent by the terminal device," as claimed. Ex. 1003,¶ 94.

The SOCKS connection request (claimed service request packet) carries a source IP address indicating the SOCKS client (claimed terminal device). *Id.*, ¶ 95. TCP/IP protocol requires all TCP/IP packets to include a header containing a source IP address indicating the device that sent the packet. *Id.* And because the SOCKS connection request is a TCP/IP packet sent over the Internet, a POSA would understand that the SOCKS connection request contains a header with the source IP address of the SOCKS client, though not shown in the Figure above. *Id.*; *see also* Ex. 1005, 1 (SOCKS is an Internet protocol), 2 (SOCKS is a framework for TCP-based client-server application), 7 (the connection established using SOCKS is a TCP connection). Thus, RFC 1928 discloses, or at least suggests, that the SOCKS connection request contains a source IP address indicating the SOCKS client (claimed terminal device). Ex. 1003, ¶ 95.

i. Koblas discloses or suggests "*the service request packet carries a terminal domain name indicating the terminal device*"

To the extent that RFC 1928 does not expressly disclose that the SOCKS connection request carries a domain name indicating the SOCKS client (claimed terminal domain name), Koblas suggests this by disclosing that the SOCKS server uses a Configuration File mapping SOCKS client domain names to corresponding permitted/denied server IP addresses. *See* Ex. 1006, 7-9; Ex. 1003, ¶¶ 96-98.



The configuration file is located on the firewall host and is used by sockd when determining whether to accept or deny requests. The file is parsed from beginning to end, with the first fully matching line returning the accessibility. The syntax of the lines in this file is as follows:

{permit | deny} <source-host> <mask> [<<dest-host> <mask> [<operator> <port>]]

source address
destination address

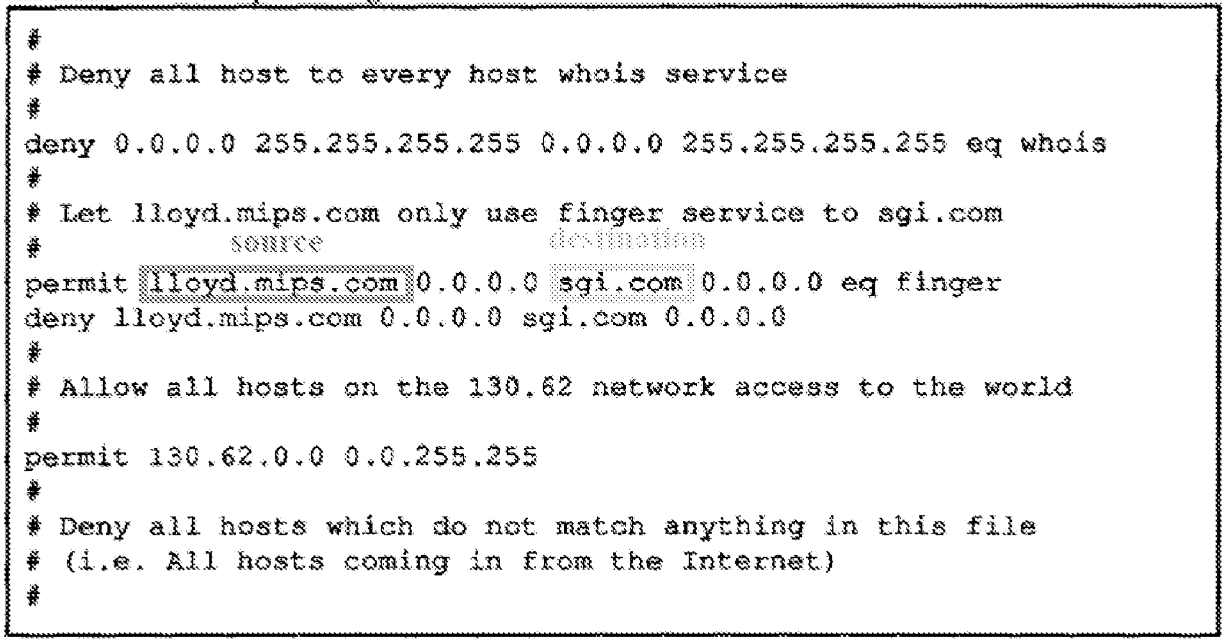
Lines begin with either 'permit' or 'deny' following which are either 2, 4, or 6 fields, containing host address and mask pairs for source and destination, as well as a boolean operator and a service port.

Ex. 1006, 7*.

As highlighted above, in the SOCKS Configuration File, the <source-host> field contains the host address of the SOCKS client source (claimed terminal device) and the corresponding <dest-host> field in the entry contains the address of the corresponding destination server. *Id.*; Ex. 1003, ¶ 97. Depending on which value the {permit | deny} field contains, the SOCKS server will respectively permit or deny the SOCKS client source in the <source-host> field to connect to the corresponding destination server address in the <dest-host> field. Ex. 1006, 7; Ex. 1003, ¶ 97. Moreover, "[h]ost addresses and services may be specified either by name or number," meaning SOCKS supports listing either a domain name or an IP address in the <source-host> and <dest-host> fields. Ex. 1006, 8; Ex. 1003, ¶ 97.

Similarly, in the Sample Configuration file in Figure 5, Koblas shows the SOCKS server permits a connection request from the source domain name lloyd.mips.com (claimed terminal domain name) to connect to the corresponding preset destination address sgi.com. Ex. 1003, ¶ 98.

FIGURE 5. A Sample Configuration File



```
#
# Deny all host to every host whois service
#
deny 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255 eq whois
#
# Let lloyd.mips.com only use finger service to sgi.com
#          source                 destination
permit lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0 eq finger
deny lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0
#
# Allow all hosts on the 130.62 network access to the world
#
permit 130.62.0.0 0.0.255.255
#
# Deny all hosts which do not match anything in this file
# (i.e. All hosts coming in from the Internet)
#
```

Ex. 1006, 8*. Because Koblas discloses that the SOCKS server screens connection requests based on the domain name of the SOCKS client source sending the connection request, Koblas at least suggests that the connection request may contain the domain name of the SOCKS client source. Ex. 1003, ¶ 98. Thus, Koblas at least suggests "*the service request packet carries a terminal domain name indicating the terminal device*," as claimed. *Id.*

ii. Rational to combine Koblas with RFC 1928/RFC 3038

It would have been obvious based on Koblas to modify the SOCKS connection request to include the domain name of the client for several reasons. *Id.*, ¶¶ 99-106. For example, Koblas provides teaching, suggestion, and/or motivation for making this modification. *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727 (2007); MPEP § 2143. Whereas RFC 1928 describes SOCKS generally to "provide a framework for client-server applications … to conveniently and securely use the services of a network firewall," Ex. 1005, 2, Koblas drills down with "several strategies which can be used to configure an Internet connection to prevent unwanted intrusion" using the SOCKS protocol, Ex. 1006, 1 (77). Thus, a POSA considering RFC 1928 and the general framework of the SOCKS protocol would have looked to Koblas for specific examples of strategies to configure secure connections using SOCKS. Ex. 1003, ¶ 99.

As discussed above, Koblas teaches that the SOCKS protocol applies access controls to SOCKS connection requests using information contained in a "Configuration File." Ex. 1006, 7-9. The Configuration File maps terminal domain names to corresponding allowed or denied server addresses. *See id.*, 8 (for example, permitting the terminal domain name "lloyd.mips.com" to access the service server domain name "sgi.com"). This means that, in at least some scenarios, the SOCKS server needs the domain name—not the IP address—of the SOCKS client to determine whether to permit or deny the connection request. Ex. 1003, ¶ 100. There are two ways the SOCKS server can determine the domain name of the SOCKS client:

The first, and most logical, way is that the SOCKS client simply provides its domain name to the SOCKS server in the connection request. *Id.*, ¶ 101. The SOCKS server needs the SOCKS client's domain name to apply the Configuration File rules for that SOCKS client to its connection request, and a POSA would have recognized having the SOCKS client simply include its domain name in a connection request as the simplest and most apparent way for the SOCKS server to learn domain name of the SOCKS client. *Id.* For this reason, a POSA would have found it obvious modify SOCKS as described in RFC 19298 to include the domain name of the SOCKS client in the connection request. *Id.*

The second way, the SOCKS server resolves the domain name of the SOCKS client from its source IP address (e.g., contained in the connection request packet). *Id.* ¶ 102. And, after resolving the domain name of the SOCKS client, the SOCKS server may then identify and apply that client's rules in the Configuration File. *Id.* But this way is less efficient than the first way, requiring the extra step of resolving the SOCKS client domain name. *Id.*

To eliminate this extra step of converting the IP address of the SOCKS client to a domain name before applying the Configuration File, it would have been obvious to simply have the SOCKS client include its domain name in the SOCKS connection request (rather than just its IP address). *Id.*, ¶ 103. This would allow the SOCKS proxy server to immediately apply the access controls in the Configuration File upon receiving a connection request, speeding up and simplifying the process of establishing the connection. *Id.* It would also eliminate the need for the SOCKS proxy server to maintain and update information mapping SOCKS client IP addresses to domain names, request services from other devices to convert client IP addresses to domain names, and/or perform other DNS-like functions to convert the source IP address into a domain name so the Configuration File can be applied. *Id.* Thus, the combination would allow simplifying the configuration of the DNS proxy server itself could and streamlining the process for establishing a secure connection. *Id.* Accordingly, a POSA would have been motivated to modify the SOCKS protocol as described in RFC 1928 to include the domain name of the SOCKS client in a connection request. *Id.*

This modification could have been made with mere routine skill in the art and a reasonable expectation of success. *Id.* ¶ 104. Indeed, a fully-qualified domain name is essentially equivalent to its corresponding IP address because both represent the same address of the same device on a network. Ex. 1003, ¶ 104; Ex. 1012 (defining "domain name" as "[a]n address of a network connection that identifies the owner of that address in a hierarchical format"). They differ mainly in their form: a domain name is easily read and memorized by a person whereas a machine-readable IP address is not. Ex. 1003, ¶ 104. The '040 Patent itself recognizes this essential equivalence between a domain name and its IP address, explaining that "mutual conversion [can be achieved] between a domain name which is readily memorized by a user and a machine recognizable IP address." Ex. 1001, 8:30-33.

Accordingly, a POSA would have viewed modifying the SOCKS connection request to include the client's domain name---when the connection request already contains that same address in its other form as an IP address—as a simple, unintrusive change. Ex. 1003, ¶ 105. For example, a POSA would have expanded the structure of the SOCKS connection request, discussed above, to include an additional client domain name field containing the domain name of the SOCKS client. *Id.* This minor change would not otherwise impact operation of SOCKS protocol and, as discussed, advantageously simplifies the process of applying the access controls in the Configuration File and related functionality. *Id.*

Accordingly, RFC 1928 and Koblas render obvious "*receiving a service request packet sent by a terminal device, wherein the service request packet carries ... a server domain name indicating the service server requested by the service request packet sent by the terminal device*," as claimed.

c. [1.2] "*resolving the received server domain name to obtain a service server Internet protocol (IP) address; and*"

RFC 1928 and RFC 3089 teach or at least suggest this element. Ex. 1003, ¶¶ 107-109. Resolving the received fully-qualified domain name of the server (claimed received server domain name) to obtain an IP address of that server (claimed service server IP address) is a basic and necessary function of SOCKS. Accordingly, although RFC 1928 does not explicitly detail this process, a POSA would understand it to be taught, or at least suggested, by RFC 1928's disclosure and teaching of the SOCKS protocol. Ex. 1003, ¶ 107; *see* also RFC 3089. RFC 1928 only lacks a detailed description on this aspect because RFC 1928 focuses on the broader framework of the SOCKS protocol and does not attempt to describe such well-known and basic aspects of the protocol. *Id.*; *see generally* Ex. 1005. Nevertheless, RFC 3089 does explicitly teach resolving the IP address of the server domain name as claimed in this limitation. Thus, it would have been obvious to do so in view of the teachings of RFC 1928 and RFC 3089. Ex. 1003, ¶ 107.

i. RFC 3089 discloses element [1.2]

RFC 3089 describes a gateway mechanism for the SOCKS proxy server to handle both IPv6 and IPv4. *See* Ex. 1007, 1 ("The SOCKS-based IPv6/IPv4 gateway mechanism is based on a mechanism that relays two 'terminated' IPv4 and IPv6 connections at the 'application layer' (<u>the SOCKS server</u>)"). RFC 3098 explains that the "characteristics [of the SOCKS server] are <u>inherited</u> from those of the connection relay mechanism at the application layer and those of the <u>native SOCKS mechanism</u>." *Id.*, 1. Thus, a POSA would have understood that the SOCKS server discussed in RFC 3089 inherits, and thus includes, the functionality of the SOCKS server described in the earlier RFC 1928. Ex. 1003, ¶ 108. That is, when RFC 3089 refers to the SOCKS server, it is the same SOCKS server referenced in RFC 1928. *Id.*

RFC 3089 describes in detail the process by which the SOCKS server resolves the fully-qualified domain name (FQDN) of the destination node (Destination D)—the claimed service server—to obtain its "real IP address"—the claimed service server IP address:

> The detailed internal procedure of the "DNS name resolving delegation" and address mapping management related issues are described as follows.
>
> 1. An application on the source node (Client C) tries to get the IP address information of the destination node (Destination D) by calling the DNS name resolving function (e.g., gethostbyname()). At this time, the logical host name ("FQDN") information of the Destination D is passed to the application's *Socks Lib* as an argument of called APIs.
> 2. Since the *Socks Lib* has replaced such DNS name resolving APIs, the real DNS name resolving APIs is not called here. The argued "FQDN" information is merely registered into a mapping table in *Socks Lib*, and a "fake IP" address is selected as information that is replied to the application from a reserved special IP address space that is never used in real communications (e.g., 0.0.0.x). The address family type of the "fake IP" address must be suitable for requests called by the applications. Namely, it must belong to the same address family of the Client C, even if the address family of the Destination D is different from it. After the selected "fake IP" address is registered into the mapping table as a pair with the "FQDN", it is replied to the application.
> 3. The application receives the "fake IP" address, and prepares a "socket". The "fake IP" address information is used as an element of the "socket". The application calls socket APIs (e.g., connect()) to start a communication. The "socket" is used as an argument of the APIs.
> 4. Since the *Socks Lib* has replaced such socket APIs, the real socket function is not called. The IP address information of the argued socket is checked. If the address belongs to the special address space for the fake address, the matched registered "FQDN" information of the "fake IP" address is obtained from the mapping table.
> 5. The "FQDN" information is transferred to the *Gateway* on the relay server (Gateway G) by using the SOCKS command that is matched to the called socket APIs. (e.g., for connect(), the CONNECT command is used.)
> 6. Finally, the real DNS name resolving API (e.g., getaddrinfo()) is called at the *Gateway*. At this time, the received "FQDN" information via the SOCKS protocol is used as an argument of the called APIs.
> 7. The *Gateway* obtains the "real IP" address from a DNS server, and creates a "socket". The "real IP" address information is used as an element of the "socket".
> 8. The *Gateway* calls socket APIs (e.g., connect()) to communicate with the Destination D. The "socket" is used as an argument of the APIs.

Ex. 1007, 5-6. Accordingly, RFC 3089 discloses "*resolving the received server domain name to obtain a service server Internet protocol (IP) address*," as claimed. Ex. 1003, ¶ 109.

ii. Rationale to Combine RFC 3089 with RFC 1928

A POSA would have found it obvious to modify RFC 1928 to resolve the received fully-qualified domain name of the server (claimed received server domain name) to obtain an IP address of that server (claimed service server IP address), as disclosed by RFC 1928. Ex. 1003, ¶¶110-113. First, the references provide teaching, suggestion, and/or motivation for making this modification. *KSR*, 127 S. Ct. 1727 (2007); MPEP § 2143. Both RFC 1928 and RFC 3089 relate to the same SOCKS protocol, so a POSA considering the general framework of SOCKS in RFC 1928 would have looked to other SOCKS references like RFC 3089 for additional implementation details about various aspects of the system. Ex. 1003, ¶ 110.

Moreover, at the outset of describing SOCKS' domain name resolution process, RFC 3089 explains that, "[i]n <u>all</u> communication applications, <u>it is [] necessary to obtain destination IP address information to start a communication</u>." Ex. 1007, 4. Ex. 1003, ¶ 111. As explained above, RFC 1928 teaches that SOCKS supports receiving the destination address as a fully-qualified <u>domain name</u> in the connection request. Ex. 1005, 4-5; Ex. 1003, ¶ 111. Thus, according to RFC 1928, it is absolutely necessary in this scenario to resolve the fully-qualified domain name into a corresponding IP address—or else the SOCKS server cannot start the connection. Ex. 1007, 4. Ex. 1003, ¶ 111. Accordingly, to implement a working SOCKS system, a POSA would have found it obvious (and necessary) to modify SOCKS as described in RFC 1928 to resolve the fully-qualified domain name of the destination server (claimed received server domain name) to obtain an IP address of the destination server (claimed service server IP address), as claimed. Ex. 1003, ¶ 111.

The combination merely involves the use of known technique (RFC 3089's resolving a fully-qualified domain name) to improve the same system (SOCKS communication system according to RFC 1928) in the same way, or applying a known technique (RFC 3089's resolving a fully-qualified domain name) to a known system (SOCKS communication system according to RFC 1928) ready for improvement to yield predictable results. *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727 (2007); MPEP § 2143. For example, since the SOCKS protocol as described in RFC 1928 would be unable to establish a connection without first resolving the fully-qualified domain name of the destination as described in RFC 3089, adding this functionality would improve SOCKS as described in RFC 1928 in the same way it operates in RFC 3089. Ex. 1003, ¶ 112. Additionally, because the combination simply adds necessary SOCKS functionality to the general

SOCKS system described in RFC 1928 to make it operational, the combination would yield predictable results. Ex. 1003, ¶ 112.

Accordingly, RFC 1928 and RFC 3089 render obvious "*resolving the received server domain name to obtain a service server Internet protocol (IP) address*," as claimed.

d. [1.3] "*discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*"

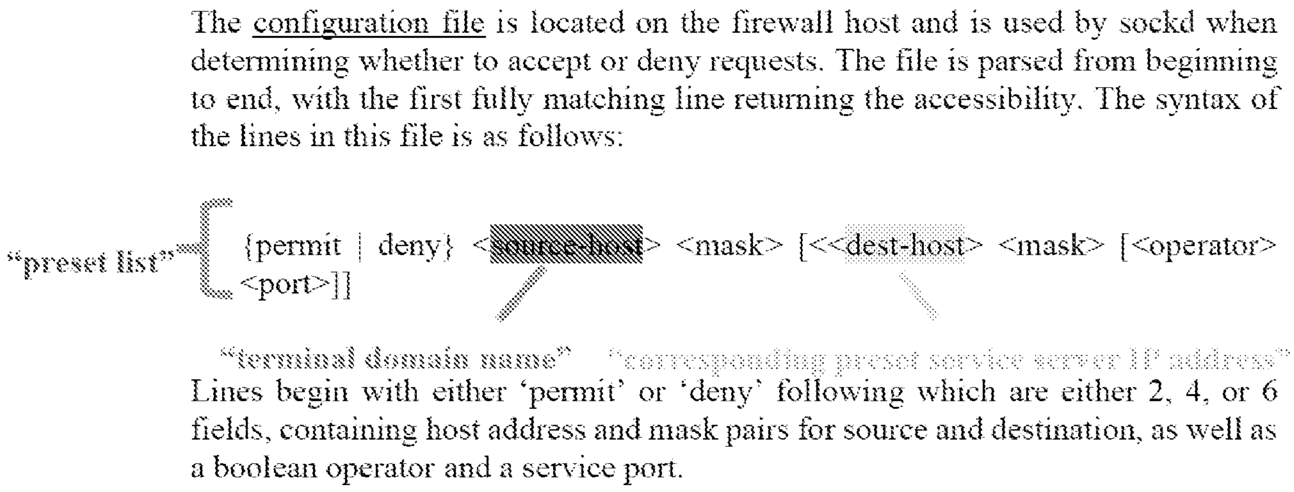
The combination of RFC 1928, Koblas, and RFC 3089 renders obvious this element. Ex. 1003. ¶¶ 114-128. As explained below, Koblas discloses or suggests discarding the SOCKS connection request (service request packet) if the resolved address of the destination server (resolved service server IP address) is not listed as an allowed address (does not belong to a preset service server IP address) corresponding to the domain name of the SOCKS client (received terminal domain name) in a Configuration File (preset list). Ex. 1003, ¶ 114. Moreover, it would have been obvious to modify the RFC 1928/RFC 3089 combination discussed above to include this element. *Id.*

i. Koblas discloses element [1.3]

RFC 1928 discloses that the SOCKS server discards the SOCKS connection request (claimed service request packet) if the requested connection is not "appropriate": [t]he SOCKS server evaluates the request, and either establishes the appropriate connection or denies it." Ex. 1005, 3. As explained above, the broadest reasonable interpretation of "discarding the service request packet" includes preventing unauthorized access to the resolved service server IP address. *Supra* Section I.B.1. Denying the connection request, as disclosed by RFC 1928, constitutes preventing unauthorized access to the resolved service server IP address. Ex. 1003, ¶ 115.

RFC 1928, however, does not expressly disclose the mechanism the SOCKS server uses to determine whether the requested connection is "appropriate." Thus, RFC 1928 does not expressly disclose discarding the SOCKS connection request "if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list," as claimed.

Koblas, however, teaches that the SOCKS server uses a "Configuration File" (claimed preset list) to evaluate and allow or deny the connection request. Ex. 1003, ¶ 117. The Configuration File contains an entry for each SOCKS client source (claimed terminal device) identifying corresponding "permit" or "deny" destination addresses (claimed service server IP address) to which the SOCKS server will respectively permit or deny connections requested by the SOCKS client source:



The configuration file is located on the firewall host and is used by sockd when determining whether to accept or deny requests. The file is parsed from beginning to end, with the first fully matching line returning the accessibility. The syntax of the lines in this file is as follows:

{permit | deny} <source-host> <mask> [<<dest-host> <mask> [<operator> <port>]]

Lines begin with either 'permit' or 'deny' following which are either 2, 4, or 6 fields, containing host address and mask pairs for source and destination, as well as a boolean operator and a service port.

Ex. 1006, 7*.

As highlighted above, in the SOCKS Configuration File, the <source-host> field contains the host address of the SOCKS client source (claimed terminal device) and the corresponding <dest-host> field in the entry contains the address of the corresponding destination server (claimed corresponding preset service server IP address). *Id.*; Ex. 1003, ¶ 118. Depending on which value the {permit | deny} field contains, the SOCKS server will respectively permit or deny the SOCKS client source in the <source-host> field to connect to the corresponding destination server address in the <dest-host> field. Ex. 1006, 7; Ex. 1003, ¶ 118. Additionally, Koblas explains that "[h]ost addresses and services may be specified either by name or number," meaning SOCKS supports listing either a domain name or an IP address in the <source-host> and <dest-host> fields. Ex. 1006, 8; Ex. 1003, ¶ 118. Accordingly, the Configuration File of Koblas discloses the claimed preset list containing a preset service server IP address <dest-host> corresponding to the received terminal domain name <source-host>. Ex. 1003, ¶ 118.

Koblas further teaches that the SOCKS server discards the SOCKS connection request if the Configuration File (claimed preset list) does not list the <dest-host> address (claimed resolved service server IP address) as a permitted connection for the <source-host> domain name (claimed does not belong to a preset service server IP address corresponding to the received terminal domain name). Ex. 1003, ¶ 119. This is because "[a]ccess is denied to all addresses which do not match anything in the configuration file." Ex. 1006, 8. Thus, if the SOCKS server does not find a particular resolved <dest-host> address listed in the Configuration File as permitted connection for the <source-host> domain name, it will deny the SOCKS connection request even though the Configuration File does not list the <dest-host> address as a denied connection. Ex. 1006, 8; Ex. 1003, ¶ 119.

In Figure 5, Koblas "shows an example of how the lines in a configuration file might appear":

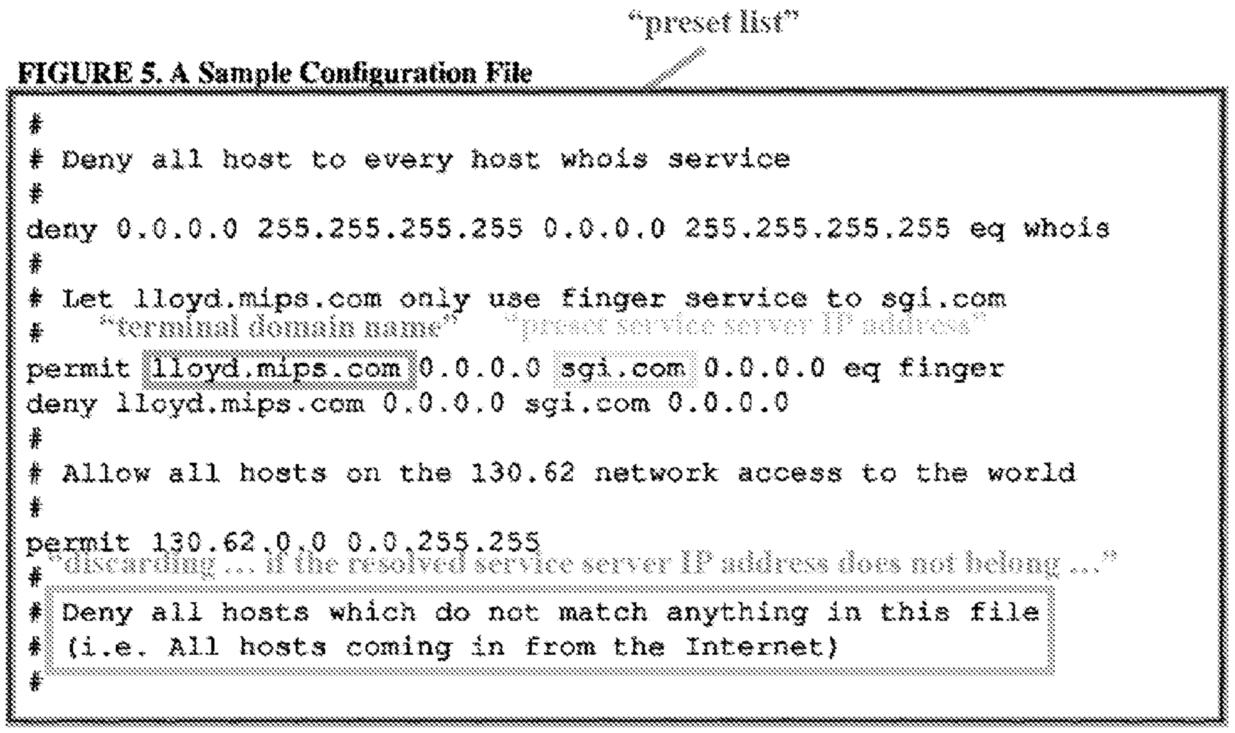

"preset list"
"terminal domain name"
"preset service server IP address"
"discarding … if the resolved service server IP address does not belong …"

FIGURE 5. A Sample Configuration File

```
#
# Deny all host to every host whois service
#
deny 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255 eq whois
#
# Let lloyd.mips.com only use finger service to sgi.com
#
permit lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0 eq finger
deny lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0
#
# Allow all hosts on the 130.62 network access to the world
#
permit 130.62.0.0 0.0.255.255
#
# Deny all hosts which do not match anything in this file
# (i.e. All hosts coming in from the Internet)
#
```

Ex. 1006, 8. In this Sample Configuration file, the SOCKS server permits a request from the source domain name lloyd.mips.com (claimed terminal domain name) to connect to the corresponding preset destination address sgi.com, and denies requested connections "which do not match anything in this file." *Id.*; Ex. 1003, ¶ 120. Although shown as a domain name in this example, the server destination address sgi.com "may be specified either by name or number," so the IP address of sgi.com could be used instead. Ex. 1006, 8; Ex. 1003, ¶ 120.

As explained above, "discarding" the service request includes preventing unauthorized access to the resolved service server IP address. *Supra* Section I.B.1. By allowing a connection if the Configuration File lists the resolved server IP address as an allowed connection for the source domain name and denying all connections that do not match anything in the Configuration File, application of the Configuration File prevents unauthorized access to the resolved server IP address. Ex. 1003, ¶ 121.

Accordingly, for at least the reasons above, Koblas discloses "*discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list*," as claimed.

ii. Rationale to combine Koblas with RFC 1928 and RFC 3089

A POSA would have found it obvious to modify SOCKS as described in RFC 1928 and/or RFC 3089 to use a Configuration File as described in Koblas to discard a SOCKS connection request if the Configuration File does not list the resolved destination address as an allowed connection for the SOCKS client. Ex. 1003, ¶¶ 123-128. The references provide teaching, suggestion, and/or motivation for making this modification. *KSR*, 127 S. Ct. 1727 (2007); MPEP § 2143. Both RFC 1928, RFC 3089, and Koblas all relate to the same SOCKS protocol, so a POSA implementing a SOCKS system as described in RFC 1928/RFC 3089 would have looked to other SOCKS references like Koblas for additional implementation details about various aspects of the system. Ex. 1003, ¶ 123.

Although RFC 1928 explains "[t]he SOCKS server evaluates the request, and either establishes the appropriate connection or denies it," RFC 1928 does not discuss a specific mechanism within SOCKS to do so because this falls outside its general scope of "provid[ing] a framework for client-server applications … to conveniently and securely use the services of a network firewall." Ex. 1005, 2. Thus, a POSA would have looked to other SOCKS references for specific mechanisms to evaluate the appropriateness of SOCKS connection requests, and Koblas's Configuration File provides one such mechanism. Ex 1003, ¶ 124.

Koblas addresses "[o]ne of the more important [security] issues" to consider when connecting to a network over the Internet: "intruders attempting to gain access to local hosts" using the SOCKS protocol. Ex. 1006, 3. A POSA would have found this solution pertinent to RFC 1928 because, in applying the Configuration File as taught by Koblas, the SOCKS server performs the function mentioned in RFC 1928: "[t]he SOCKS server evaluates the request, and either establishes the appropriate connection or denies it." Ex. 1005, 3; Ex. 1003, ¶ 125. Thus, a POSA would have found it obvious and been motivated to add the SOCKS Configuration File functionality described in Koblas to the SOCKS system as described in RFC 1928 to make SOCKS work as RFC 1928 intends. Ex. 1003, ¶ 125.

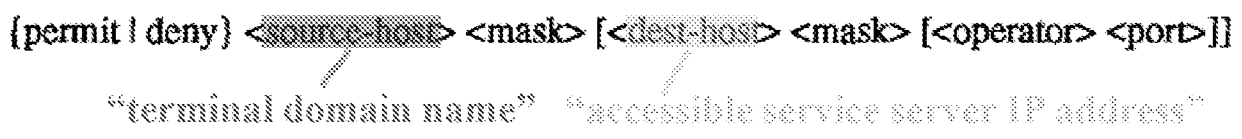
Moreover, the combination merely involves the use of a known technique (Koblas's Configuration File) to improve the same system (SOCKS system according to RFC 1928) in the same way, or applying a known technique (Koblas's Configuration File) to a known system (SOCKS system according to RFC 1928) ready for improvement to yield predictable results. *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727 (2007); MPEP § 2143. The combination would improve SOCKS as described in RFC 1928 because, as discussed, RFC 1928 teaches that the SOCKS server evaluates a connection request to confirm its appropriateness before establishing the connection, but does not expressly describe a mechanism for doing so. Ex. 1005, 2; Ex. 1003, ¶ 126. Koblas fills this gap with its Configuration-File solution for evaluating SOCKS connection requests. Ex. 1003, ¶ 126. Thus, combining Koblas's Configuration File technique with RFC 1928 would improve the SOCKS system as described in RFC 1928 in the same way it works in Koblas—preventing intruders from gaining access to local hosts by only allowing connections from certain sources to certain destinations. *Id.*

Additionally, the combination would yield predictable results and would be made through only routine skill in the art. Ex. 1003, ¶ 127. Indeed, the combination simply adds a SOCKS security solution from Koblas to the general SOCKS system described in RFC 1928. *Id.* With the Configuration File functionality added to RFC 1928, the combined SOCKS system would operate in the way RFC 1928 describes: the SOCKS server would evaluate a connection request and establish the connection if appropriate or deny it otherwise. Ex. 1005, 3; Ex. 1003, ¶ 127. Moreover, Koblas explains "[t]he configuration file is located on the firewall host and is used by sockd when determining whether to accept or deny requests." Ex. 1006, 7. By firewall, Koblas is referring to the SOCKS server. Ex. 1003, ¶ 127. Thus, a POSA would have recognized that, in combining Koblas with RFC 1928, the Configuration File of Koblas would be stored on the SOCKS server in RFC 1928 and applied when a connection request is received. *Id.* Because Koblas and RFC 1928 both describe the same SOCKS protocol, a POSA would have modified the SOCKS server in RFC 1928 to include the Configuration-File functionality of Koblas through routine skill and had a reasonable expectation of success in doing so. *Id.* In fact, most, if not all, SOCKS implementations before the priority date of the '040 Patent, and to this date, use a Configuration File that provides similar if not identical features as disclosed by Koblas. *Id.*

Accordingly, for at least the reasons above, RFC 1928, Koblas, and RFC 3089 render obvious "*discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*" as claimed.

e. [1.4] "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device.*"

The combination of RFC 1928, Koblas, and RFC 3089 renders obvious this element. Ex. 1003. ¶¶ 128-131. As highlighted below, in Koblas's Configuration File (preset list), the domain name of each SOCKS client address (terminal domain name) is provided with an accessible destination server address (accessible service server IP address) under an access authority of the SOCKS client source (terminal device):



```
{permit | deny} <source-host> <mask> [<dest-host> <mask> [<operator> <port>]]
```

"terminal domain name"
"accessible service server IP address"

Ex. 1006, 7.

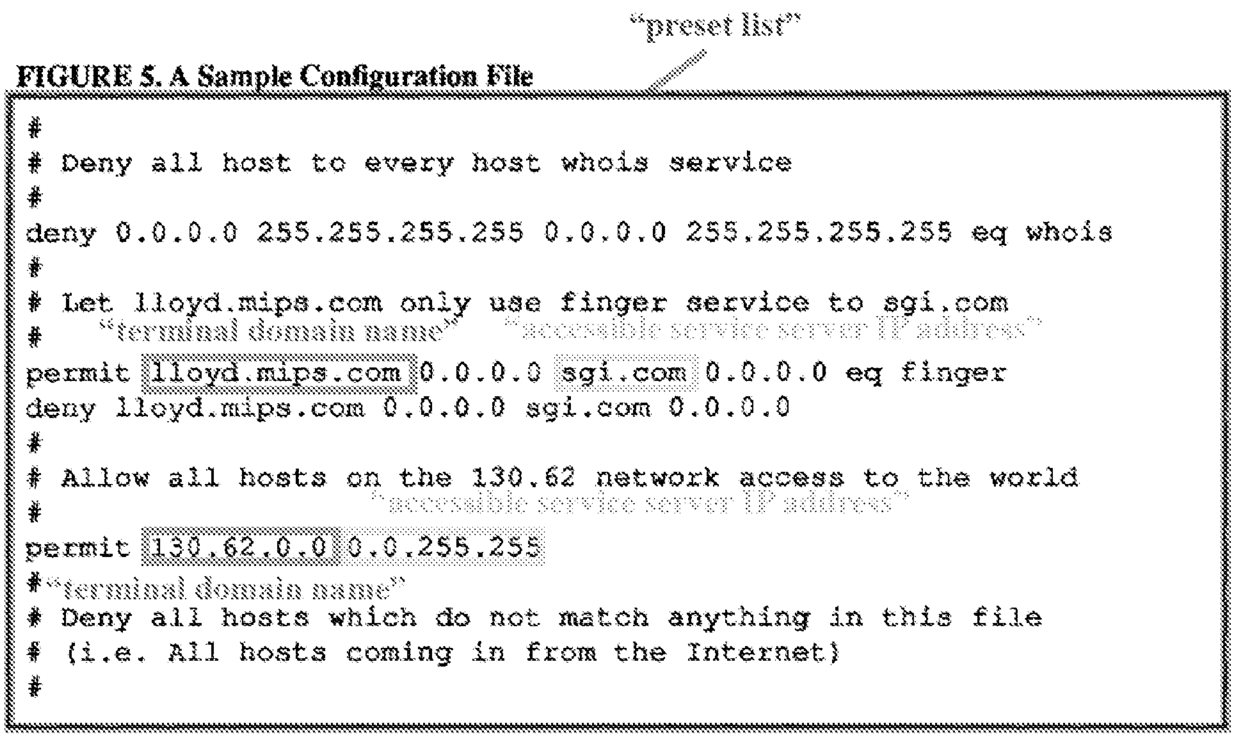

"preset list"
"terminal domain name"
"accessible service server IP address"
"accessible service server IP address"
"terminal domain name"

FIGURE 5. A Sample Configuration File

```
#
# Deny all host to every host whois service
#
deny 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255 eq whois
#
# Let lloyd.mips.com only use finger service to sgi.com
#
permit lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0 eq finger
deny lloyd.mips.com 0.0.0.0 sgi.com 0.0.0.0
#
# Allow all hosts on the 130.62 network access to the world
#
permit 130.62.0.0 0.0.255.255
#
# Deny all hosts which do not match anything in this file
# (i.e. All hosts coming in from the Internet)
#
```

Ex. 1006, 8. As explained above, the Configuration File can specify source and destination addresses "either by name or number." Ex. 1006, 8. Thus, although the Sample Configuration File above shows an IP address (e.g., 130.62.0.0) for some of the SOCKS client sources (claimed terminal device), Koblas teaches that a domain name could be used instead. Ex. 1003, ¶ 129. Thus, the combination renders obvious "the terminal domain name of each terminal device" is provided with a corresponding accessible service server IP address, as claimed.

Moreover, though Koblas only shows one accessible (i.e., permitted) destination address for each SOCKS client, it would have been obvious to include additional "permit" entries listing additional accessible IP addresses for each SOCKS client Ex. 1003, ¶ 130. For example, the Sample Configuration File only permits the SOCKS client lloyd.mips.com to access a single server address—sgi.com. *Id.* Of course, in practice, this SOCKS client and/or its user may need to access more than just one site on the network. *Id.* In this case, it would have been obvious for the administrator to add more "permit" entries to the Configuration file listing more accessible server addresses for the SOCKS client lloyd.mips.com. *Id.*

Accordingly, for at least the reasons above, RFC 1928, Koblas, and RFC 3089 render obvious "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an* *access authority of the terminal device*," as claimed.

2. Dependent Claim 4

a. [4.1] "*The method according to claim 1, wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:*

***if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device.*"**

Claim 4 depends from independent claim 1 and recites the additional elements quoted above. The combination of RFC 1928, Koblas, and RFC 3089 renders obvious independent claim 1 for the reasons discussed above. *Supra* Section II.A.1. Moreover, the combination renders obvious the additional elements of claim 4. Ex. 1003, ¶¶ 132-133.

For example, RFC 3038 discloses "[t]he SOCKS server evaluates the request, and either establishes the appropriate connection or denies it." Ex. 1005, 3. Additionally, as discussed, in the combined SOCKS system, the SOCKS server establishes a connection request if the Configuration File lists the resolved IP address of the destination server as a "permitted" connection for the domain name of the particular SOCKS client making the connection request. *See, e.g.*, Ex. 1006, 7-8. Accordingly, The combination of RFC 1928, Koblas, and RFC 3089 renders obvious "*wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises: if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device*," as recited in claim 4. Ex. 1003, ¶ 133

3. Dependent Claim 5

a. [5.1] "*The method according to claim 1, wherein, after the resolving the received server domain name to obtain the service*

> ***server Internet protocol (IP) address, the method further comprises:***
>
> ***if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device.***"

Claim 5 depends from independent claim 1 and recites the additional elements quoted above. The combination of RFC 1928, Koblas, and RFC 3089 renders obvious independent claim 1 for the reasons discussed above. *Supra* Section II.A.1. Moreover, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious additional elements of claim 5. Ex. 1003, ¶¶ 134-137.

For example, in Koblas, the Configuration File specifies that the SOCKS client domain name lloyd.mips.com may access the server sgi.com, but only for purposes of using the "finger service" provided by that server. Ex. 1006, 8; Ex. 1003, ¶ 135. This is reflected in a comment in the Configuration File above the permit and deny entries for lloyd.mips.com, stating "Let lloyd.mips.com <u>only use finger service to sgi.com</u>." Ex. 1006, 8; *see also id.*, Ex. 1003, ¶ 135. Accordingly, the "permit" entry for lloyd.mips.com specifies that lloyd.mips.com can only use sgi.com's service "eq finger", and the "deny" entry beneath it denies all other connections sought by lloyd.mips.com to the server sgi.com. Ex. 1006, 8; Ex. 1003, ¶ 135.

Thus, when the SOCKS client lloyd.mips.com sends a connection request for sgi.com, the SOCKS server applies the Configuration File to determine that sgi.com is an accessible address for lloyd.mips.com. Ex. 1006, 8; Ex. 1003, ¶ 136. This discloses determining "if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list," as claimed. Ex. 1003, ¶ 136. If this is the case, the SOCKS server then determines that the SOCKS client is seeking access only to the "finger service" of sgi.com, and not to the server more broadly, before establishing the connection. Ex. 1006, 8; Ex. 1003, ¶ 136. This discloses "determining a service type of the service request according to the terminal domain name of the terminal device," as claimed. Ex. 1003, ¶ 136.

Accordingly, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious "*wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises: if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain* *name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device*," as recited in claim 5.

4. Independent Claim 6

Independent claim 6 is an apparatus claim to a "deep packet inspection (DPI) device" configured to perform a method including elements [6.pre] to [6.4], which are virtually identical to respective elements [1.pre] to [1.4] of independent claim 1. *Compare* '040 Patent claim 1 *with id.*, claim 6. As set forth below, combination of RFC 1928, Koblas, and RFC 3089 renders the elements of independent claim 6 for reason similar to those discussed above for independent claim 1.

a. [6.pre] "*A deep packet inspection (DPI) device comprising a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising:*"

To the extent the preamble is limiting, the SOCKS server described in RFC 1928, Koblas, and RFC 3089 discloses the claimed DPI device. Ex. 1003, ¶ 139. Moreover, since a server is a computer, it would be understood to have a non-transitory computer readable storage medium (e.g., memory), instructions stored in memory for performing the functions attributed to the SOCKS server in the references, and one or more processors that execute the instructions to perform those functions. *Id.*

Accordingly, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious "*[a] deep packet inspection (DPI) device comprising a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method*," as claimed.

b. [6.1] "*receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;*"

See the discussion above for element [1.1]. *Supra* Section II.A.1.b.

c. [6.2] "*resolving the server domain name to obtain a service server Internet protocol (IP) address; and*"

See the discussion above for element [1.2]. *Supra* Section II.A.1.c.

d. **[6.3] "*discarding the packet if the service server IP address resolved does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*"**

See the discussion above for element [1.3]. *Supra* Section II.A.1.d.

e. **[6.4] "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with accessible service server IP addresses under an access authority of the terminal device.*"**

See the discussion above for element [1.4]. *Supra* Section II.A.1.e.

5. Dependent Claim 9

a. **[9.1] "*The DPI device according to claim 6, wherein after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:***

***if the service server IP address resolved belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device.*"**

Claim 9 depends from independent claim 6 and recites the additional elements quoted above. The combination of RFC 1928, Koblas, and RFC 3089 renders obvious independent claim 6 for the reasons discussed above. *Supra* Section II.A.4. Moreover, claim 9 recites subject matter virtually identical to that discussed above for dependent claim 4. Accordingly, for the reasons discussed above in connection with claim 4, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious the subject matter of claim 9. *Supra* Section II.A.2.

6. Dependent Claim 10

a. **[10.1] "*The DPI device according to claim 6, wherein after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:***

> ***if the service server IP address resolved belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device.*"**

Claim 10 depends from independent claim 6 and recites the additional elements quoted above. The combination of RFC 1928, Koblas, and RFC 3089 renders obvious independent claim 6 for the reasons discussed above. *Supra* Section II.A.4. Moreover, claim 10 recites subject matter virtually identical to that discussed above for dependent claim 5. Accordingly, for the reasons discussed above in connection with claim 5, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious the subject matter of claim 10. *Supra* Section II.A.3.

7. Independent Claim 11

Independent claim 11 is a system claim to a "deep packet inspection (DPI) device" and a "terminal device." As explained above, the SOCKS server and SOCKS client in RFC 1928, Koblas, and RFC 3089 respectively correspond to the claimed DPI device and terminal device. *Supra* Sections II.A.4. Substantively, claim 11 is virtually identical to independent claims 1 and 6 discussed above. Accordingly, as set forth below, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious the elements of independent claim 11 for reasons similar to those discussed above for independent claims 1 and 11.

a. [11.pre] "*A system, comprising: a deep packet inspection (DPI) device; and a terminal device:*"

To the extent the preamble is limiting, the combination of RFC 1928, Koblas, and RFC 3089 renders obvious this element. *Supra* Sections II.A.4.a, 1.a.

b. [11.1] [*a terminal device*] "*configured to send a service request packet to the DPI device, wherein the packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request sent by the terminal device;*" and

> ***"the DPI device having a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising: receiving the service request packet sent by the terminal device;"***

See the discussion above for element [1.1]. *Supra* Sections II.A.4.b, 1.b.

c. **[11.2] "*resolving the server domain name to obtain a service server Internet protocol (IP) address; and*"**

See the discussion above for element [1.2]. *Supra* Section II.A.1.c.

d. **[11.3] "*discarding the packet if the service server IP address resolved does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*"**

See the discussion above for element [1.3]. *Supra* Section II.A.1.d.

e. **[11.4] "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with accessible service server IP addresses under an access authority of the terminal device.*"**

See the discussion above for element [1.4]. *Supra* Section II.A.1.e.

B. Grounds 2 and 3: Treuhaft Anticipates and/or Renders Obvious Claims 1, 4-6, and 9-11 of the '040 Patent Under §§ 102 and/or 103, Respectively

Treuhaft anticipates and/or renders obvious claims 1, 4-6, and 9-11 of the '040 Patent under §§ 102 and/or 103, respectively.

1. Independent Claim 1

a. **[1.pre] "*A packet receiving method, comprising:*"**

To the extent the preamble is limiting, Treuhaft discloses a packet receiving method. Ex. 1003, ¶¶ 154-155. Generally, in Treuhaft, a host device 105 sends a DNS query 110 to a DNS name server 120. Ex. 1008, ¶¶ [0025]-[0027], [0030]-[0032]. The DNS query 110 requests the DNS name server 102 to resolve a domain name, contained in the DNS query, into a corresponding IP address and return the resolved IP address to the host device 105. *Id.* In response to receiving the DNS query 110, the DNS name server 102 performs the steps shown in FIGS. 5B and 5C to resolve the domain name into a corresponding IP address and return the IP address to the host device 105 in a DNS response 170. *Id.*, ¶¶ [0027], [0032], [0064]-[0067]. The claimed terminal device, packet(s), and packet receiving method are highlighted in Figures 1, 5B, and 5C of Treuhaft below:

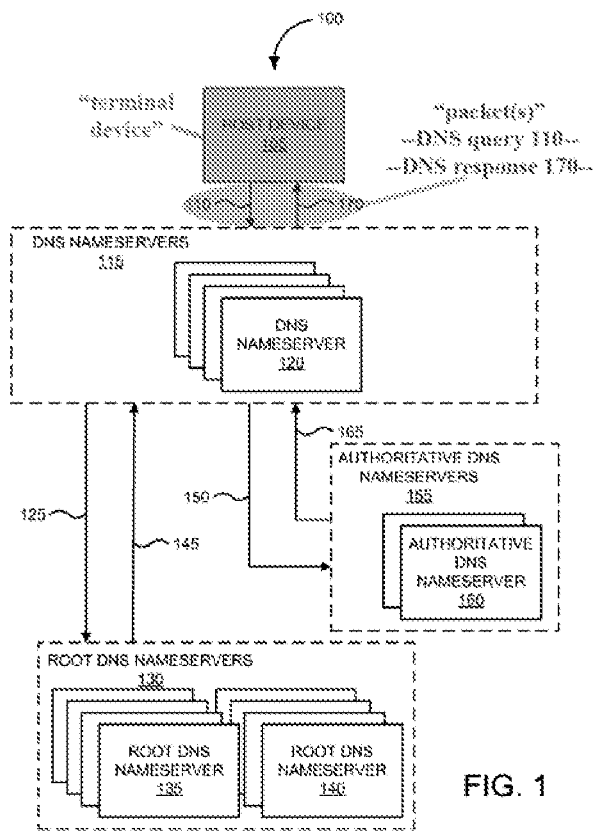


FIG. 1

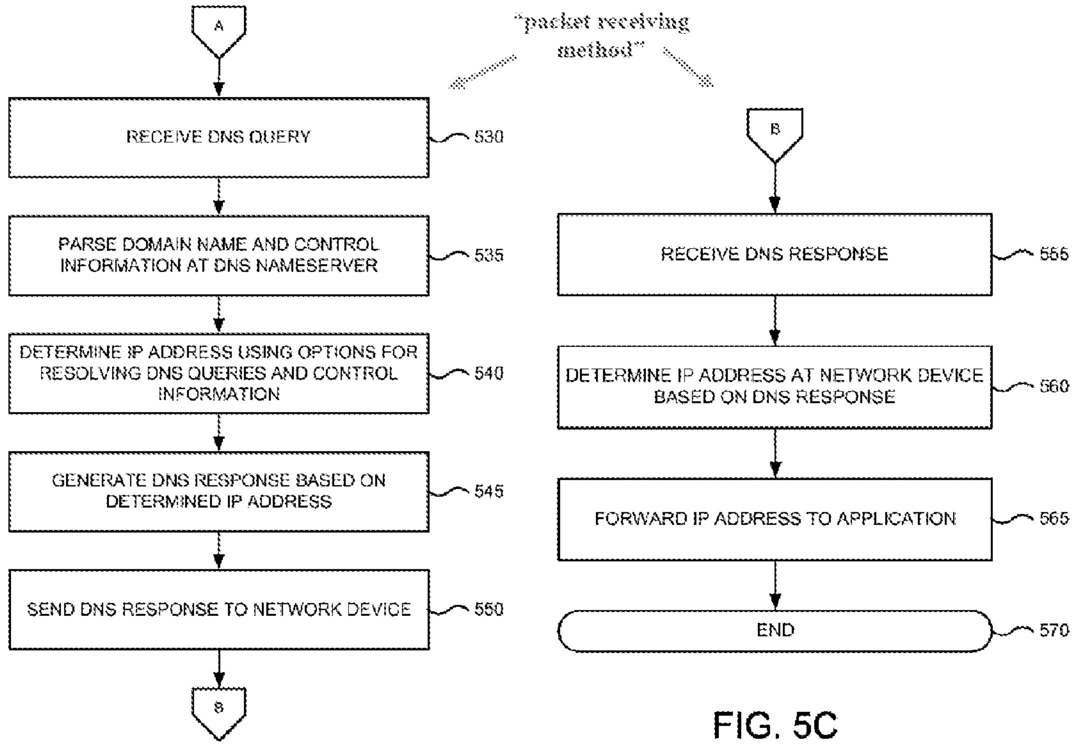

A
"packet receiving method"
RECEIVE DNS QUERY
530
PARSE DOMAIN NAME AND CONTROL INFORMATION AT DNS NAMESERVER
535
DETERMINE IP ADDRESS USING OPTIONS FOR RESOLVING DNS QUERIES AND CONTROL INFORMATION
540
GENERATE DNS RESPONSE BASED ON DETERMINED IP ADDRESS
545
SEND DNS RESPONSE TO NETWORK DEVICE
550
B
B
RECEIVE DNS RESPONSE
555
DETERMINE IP ADDRESS AT NETWORK DEVICE BASED ON DNS RESPONSE
560
FORWARD IP ADDRESS TO APPLICATION
565
END
570

FIG. 5B

FIG. 5C

Treuhaft, FIGS. 1, 5B, 5C*

Accordingly, Treuhaft discloses "*a packet receiving method*," as claimed.

b. [1.1] "*receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;*"

Treuhaft discloses, or at least suggests, this element. Ex. 1003, ¶¶ 156-169. Specifically, as explained below, Treuhaft discloses receiving a service request packet (modified DNS query 110) sent by a terminal device (host device 105). *Id.* Additionally, the service request packet (modified DNS query 110) carries a terminal domain name (control information) indicating the terminal device (host device 105) and a server domain name (domain name) indicating the service server requested by the service request packet (modified DNS query 110) sent by the terminal device (host device 105). *Id.*

In step 525 of FIG. 5A, the host device 105 sends a modified DNS query to the DNS name server 120, and the DNS name server 120 receives the modified DNS query in step 530 of FIG. 5B. Ex. 1008, ¶ [0063] ("In step 525, the modified DNS query is sent to a DNS nameserver. For example, the modified DNS query may be sent to DNS nameserver 120."), ¶ [0064] ("in step 530, the DNS query is received"). Treuhaft's DNS name server 120 receiving the modified DNS query from the host device 105 corresponds to the claimed receiving a service request packet sent by a terminal device. Ex. 1003, ¶ 157.

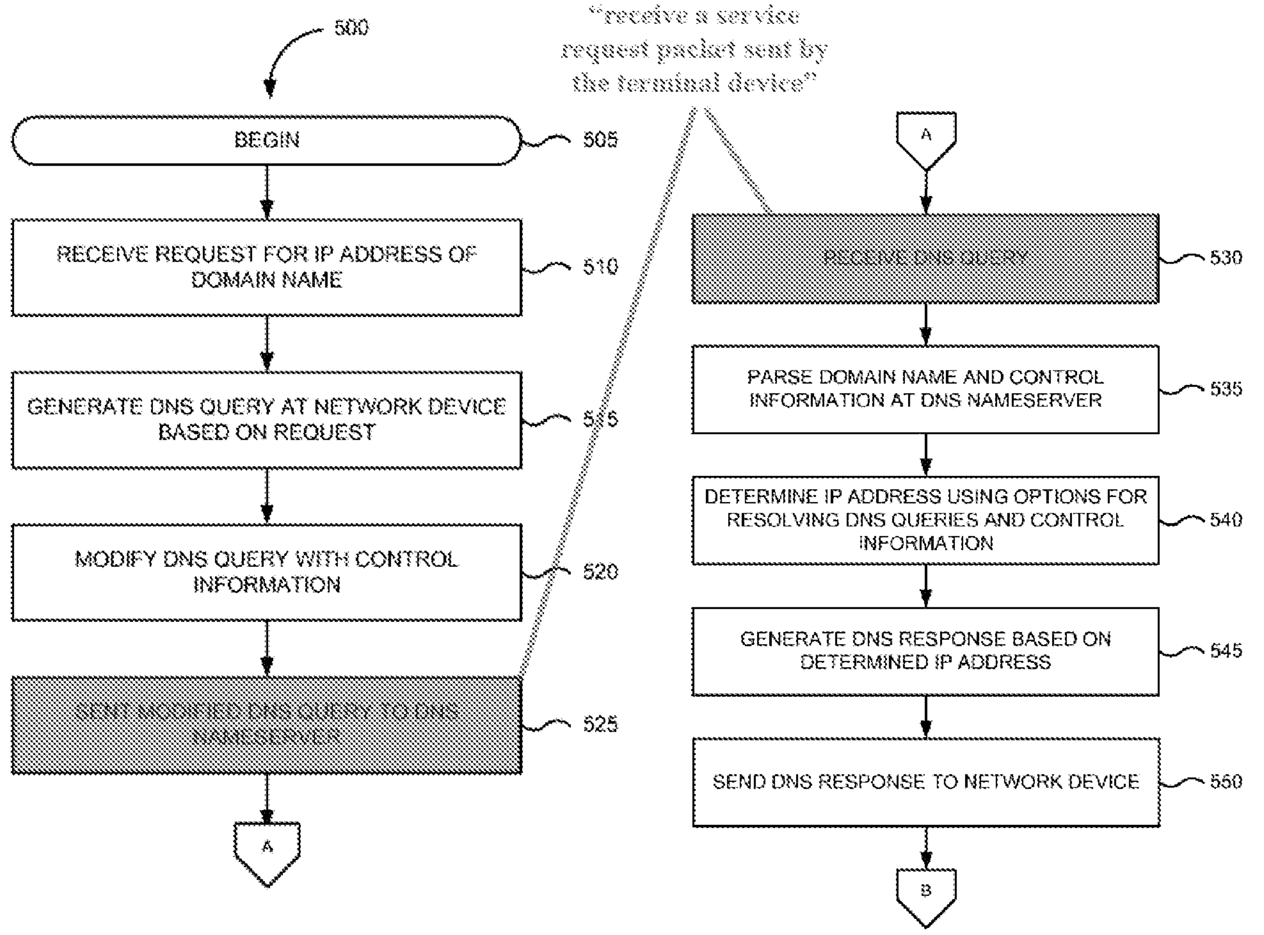
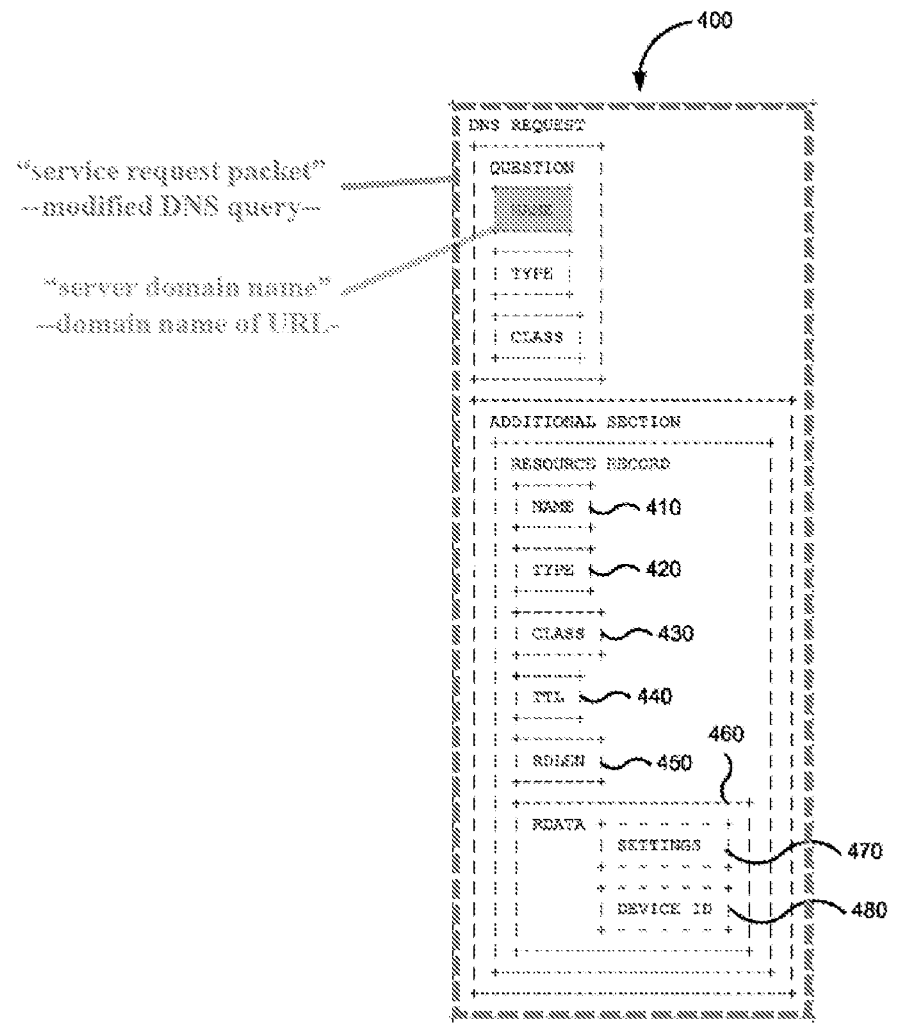

"receive a service request packet sent by the terminal device"
500
BEGIN
505
RECEIVE REQUEST FOR IP ADDRESS OF DOMAIN NAME
510
GENERATE DNS QUERY AT NETWORK DEVICE BASED ON REQUEST
515
MODIFY DNS QUERY WITH CONTROL INFORMATION
520
SENT MODIFIED DNS QUERY TO DNS NAMESERVER
525
A
A
RECEIVE DNS QUERY
530
PARSE DOMAIN NAME AND CONTROL INFORMATION AT DNS NAMESERVER
535
DETERMINE IP ADDRESS USING OPTIONS FOR RESOLVING DNS QUERIES AND CONTROL INFORMATION
540
GENERATE DNS RESPONSE BASED ON DETERMINED IP ADDRESS
545
SEND DNS RESPONSE TO NETWORK DEVICE
550
B

FIG. 5A FIG. 5B

Treuhaft's DNS modified DNS query discloses "a service request packet … carr[ying] … a server domain name indicating a service server required by the service request packet sent by the terminal device," as claimed. Ex. 1003, ¶ 158. Treuhaft explains that "host device 105 makes DNS query 110, for example for the IP address of the domain name 'www.cnet.com,' to a set of DNS nameservers 115." Ex. 1008, ¶ [0025]. That is, in this example, the DNS query contains the domain name "www.cnet.com" of a server host device 105 seeks to access, and host device 105 needs the server's IP address from the DNS name server to do so. Ex. 1003, ¶ 158. In Figure 2, Treuhaft shows an example format of the DNS query 400:



Treuhaft, FIG. 4*

As highlighted in Figure 4 above, the DNS query 400 includes a NAME field containing the domain name of the URL the host device 105 seeks to access. Ex. 1008, ¶¶ [0054]; Ex. 1003, ¶ 158. Thus, like the '040 Patent's service request packet, the DNS query of Treuhaft "carries … a server domain name indicating a service server required by the service request." Ex. 1001, 3: 27-31; *see also id.*, FIG. 4 (illustrating the claimed service request packet as an HTTP request containing a domain name).

Under the broadest reasonable interpretation standard, the modified DNS query of Treuhaft is a "packet" because is a unit of information transmitted as a whole: a DNS protocol request message. *See* Ex. 1008, ¶¶ [0041], [0043] (alternatively referring to the DNS query as a "DNS message"); Ex. 1012, 5 (definition of "packet"); Ex. 1003, ¶ 159. A DNS query is a self-contained unit of information transmitted as a whole, in a format required by DNS protocol. Ex. 1003, ¶ 159. Similarly, the '040 patent provides an example in which the service request packet is an HTTP GET message—a self-contained unit of information transmitted as a whole according to HTTP. Ex. 1001, FIG. 2 (showing the format of the HTTP service request message); Ex. 1003, ¶ 159.

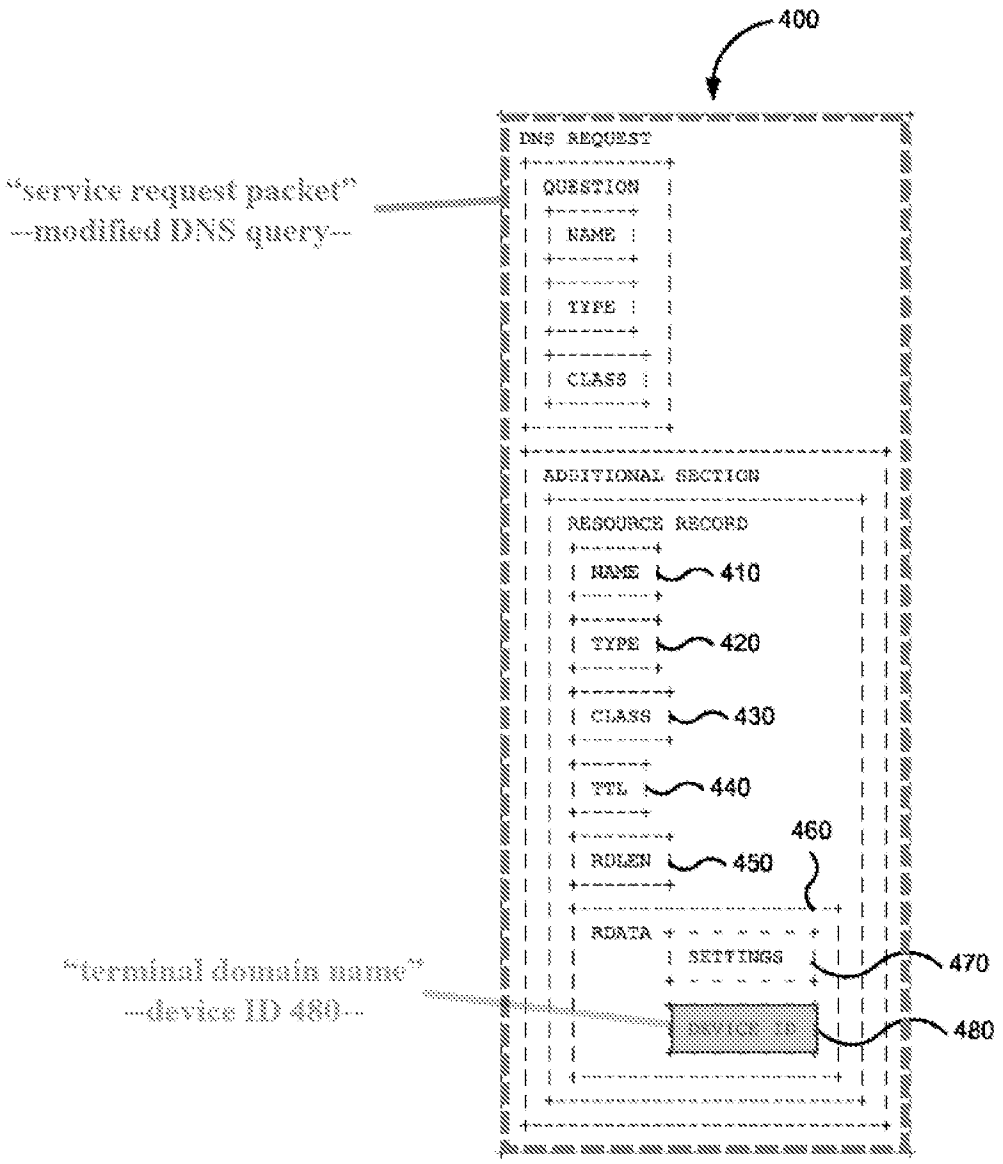
Accordingly, Treuhaft discloses, or at least suggests, *"receiving a service request packet sent by a terminal device, wherein the service request packet carries ... a server domain name indicating the service server requested by the service request packet sent by the terminal device,"* as claimed.

Treuhaft further discloses or renders obvious that the modified DNS query (claimed service request packet) also carries a terminal domain name indicating the terminal device (host device 105), as claimed. Ex. 1003, ¶ 161. Specifically, before the host device 105 sends the DNS query to name server 102, "[i]n step 520, the DNS query is modified with control information." Ex. 1008, ¶ [0058]; *see also id.*, ¶ [0067] ("control information may be encoded into an individual DNS query that enables a DNS nameserver to identify DNS resolution options, filters, or features to apply when resolving the individual DNS query"). This control information included in the modified DNS query discloses, or at least suggests, the claimed terminal domain name. Ex. 1003, ¶ 161.

Under the broadest reasonable interpretation, "terminal domain name indicating the terminal device" includes an identifier associated with an owner of the terminal device. *See* Ex. Ex. 1012, 4 (definition of "domain name"); Ex. 1001, 3: 32-51 (describing a terminal domain name as a "unique identifier" of the terminal device); Ex. 1003, ¶ 162. Treuhaft discloses several examples in which the control information serves as an identifier associated with an owner of the terminal device: "[t]he control information may specify ... a user or subscriber identifier, a device identifier, or the like." Ex. 1008, ¶ [0036]; Ex. 1003, ¶ 162. Each of the user identifier, subscriber identifier, or device identifier serves as an identifier of the owner of the terminal device—in Treuhaft's case, host device 105—and thus discloses or suggests the claimed terminal domain name. Ex. 1003, ¶ 162. Indeed, a domain name is "like" a user identifier, subscriber identifier, or device identifier in that it is a name or label of the device with which it is associated and/or its owner. *Id.* Thus, Treuhaft at least contemplates a domain name in its description of the control information. *Id.*

Treuhaft uses "user or subscriber" interchangeably to refer to "a user or subscriber of the OpenDNS service [who] set[s] one or more preferences or selections for how the options are to be enabled or otherwise applied when DNS nameserver 120 resolves DNS queries associated with the user," Ex. 1008, ¶ [0028]; *see also id.*, ¶¶ [0008], [0024], [0027], [0029], [0036], [0060]; Ex. 1003, ¶ 163. And when the host device 105 attempts to access a server at a certain domain name, the DNS name server 120 applies the corresponding user's preferences or selections when resolving the DNS query for the address of that domain name in step 540 of Figure 5B. *See* Ex. 1008, ¶¶ [0065]; *see also id.*, ¶¶ [0028], [0035], [0039]; Ex. 1003, ¶ 163. Thus, Treuhaft's user or subscriber information, included in the modified DNS query as control information, discloses the claimed terminal domain name because it identifies an owner associated with the host device 105 who is controlling the host device 105's access to the Internet. Ex. 1003, ¶ 163.

Similarly, the device identifier of Treuhaft alternatively or additionally discloses or suggests the claimed terminal domain name. *Id.*, ¶ 164. As highlighted in Figure 4 below, Treuhaft discloses the modified DNS query 400 may contain a device ID 480 "provided in the additional section of [the modified] DNS query." Ex. 1008, ¶ [0053].

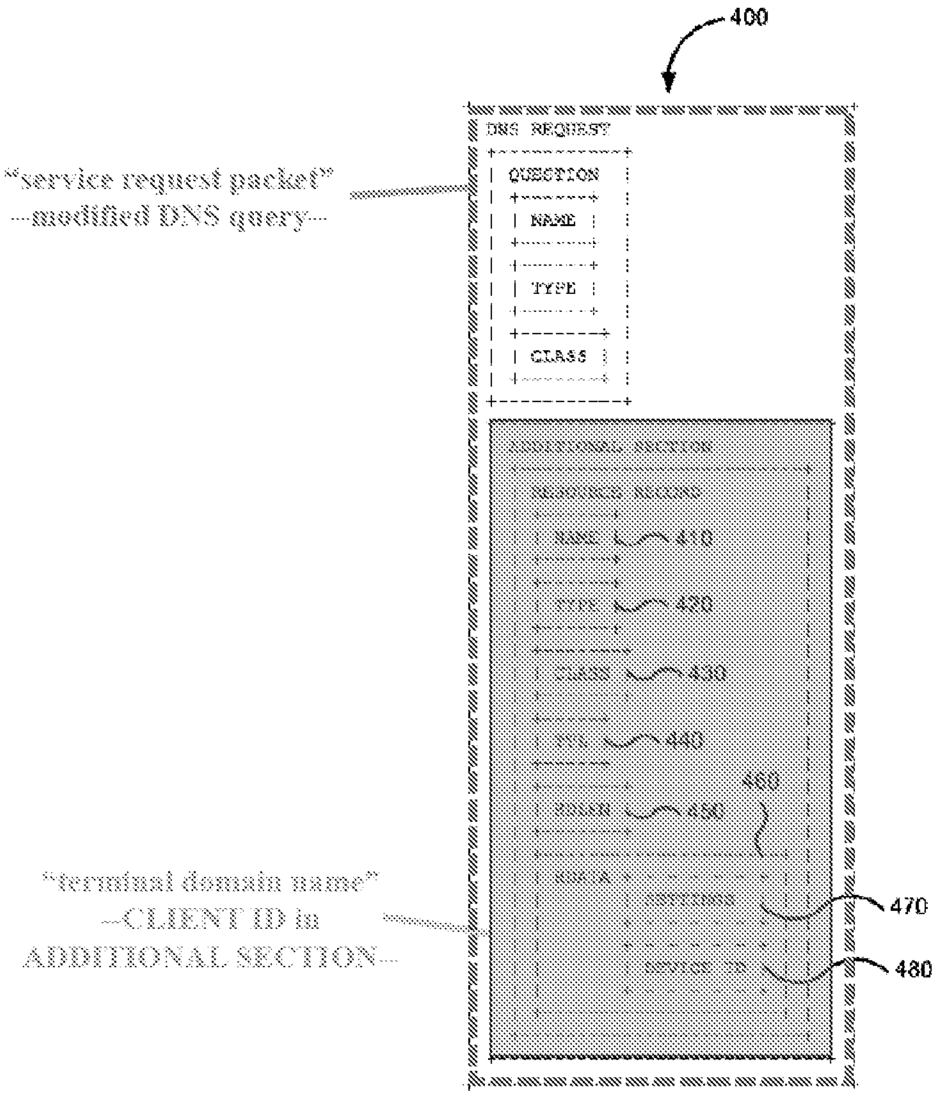


Treuhaft, FIG. 4*

The device ID 480 likewise corresponds to an owner associated with the host device 105 because "host device 105 can supply a device ID to DNS nameserver 120 by including DEVICE ID 480." *Id.*; Ex 1003, ¶ 164. That is, the host device supplies its own device ID 480 when forming the modified DNS query 400. Ex. 1003, ¶ 164. And, as explained above, the host device 105 seeking access to a certain domain name has an associated owner or subscriber to the DNS service who has supplied "one or more preferences or selections for how the options are to be enabled or otherwise applied when DNS nameserver 120 resolves DNS queries associated with the user." Ex. 1008, ¶ [0028]; *see also id.*, ¶¶ [0008], [0024], [0027], [0029], [0036], [0060]; Ex. 1003, ¶ 164.

Accordingly, Treuhaft's device ID 480, included in the modified DNS query as control information, also discloses or suggests the claimed terminal domain name because it identifies an owner associated with the host device 105 who is controlling the host device 105's access to the Internet. Ex. 1003, ¶ 164.

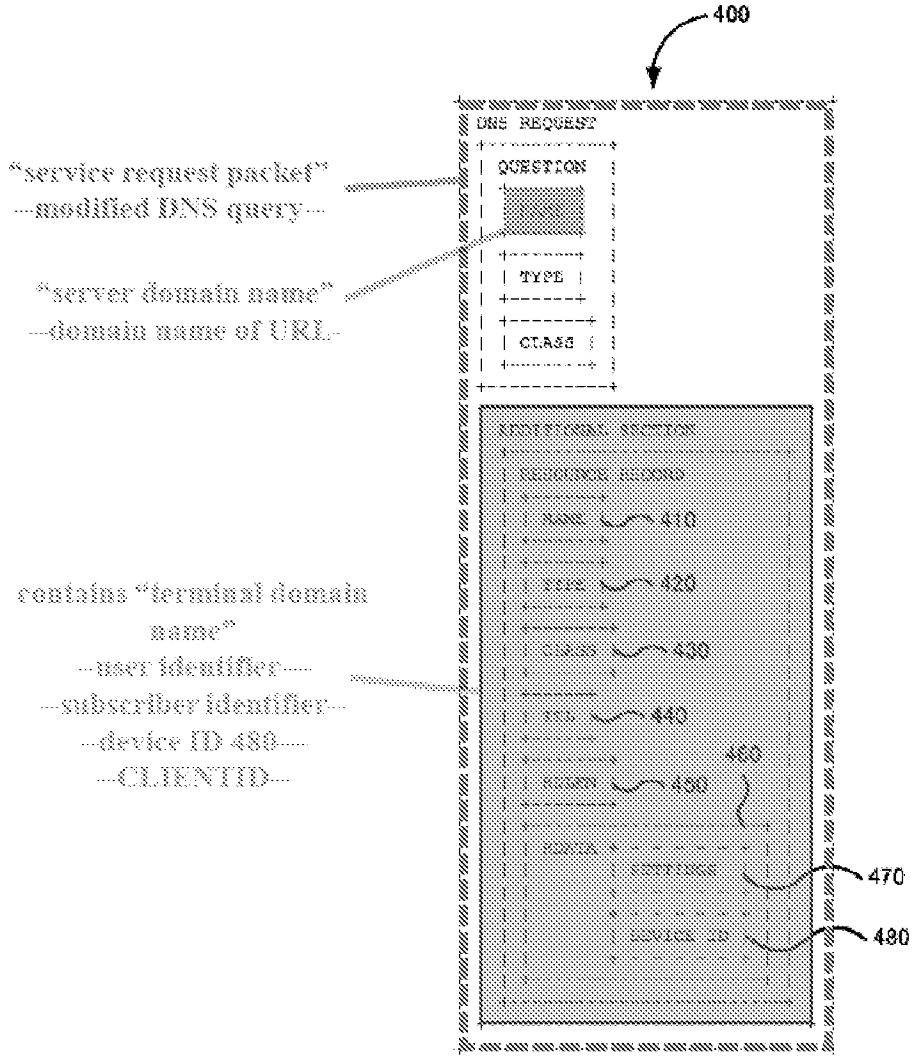
Another example disclosing or suggesting claimed terminal domain name, highlighted in Figure 4 below, Treuhaft teaches that the modified DNS query may contain a "CLIENTID for control of user, device, or vendor-specific DNS server behavior" within "the additional data section of a request." Ex. 1008, ¶ [0042]; *see also id.*, ¶¶ [0043]-[0045]; Ex. 1003, ¶ 165.



Treuhaft, FIG. 4*

Similar to the other control information discussed above, Treuhaft also uses the CLIENTID to identify the user or subscriber associated with the host device 105 and control the host device's access to the Internet accordingly. *See* Ex. 1008, ¶¶ [0042]-[0045]; Ex. 1003, ¶ 165.

Accordingly, as outlined above, the modified DNS query of Treuhaft discloses the claimed service request packet as follows. Ex. 1003, ¶ 166.



Treuhaft, FIG. 4*

To the extent it is argued Treuhaft's control information (e.g., user identifier, subscriber identifier, device ID, or CLIENTID) does not disclose a terminal domain name in the narrow sense of a hierarchical domain name in the form server.organization.type, *see* Ex. 1012, 4 (definition of domain name), it would have been obvious to a POSA to modify Treuhaft to use such a hierarchical domain name of the host device 105 as the control information. Ex. 1003, ¶ 167. Treuhaft teaches that the purpose of the host device identifier contained in the DNS query is to "enable[] the domain name service to retrieve subscriber information" which "include[s] preferences or other settings for how a user or subscriber wishes to control domain name resolution within the DNS resolution features." Ex. 1008, ¶ [0060]; *see also id.*, ¶¶ [0039] ("host device 105 may encode within a DNS query an identifier, such as an account ID or index, that specifies where DNS nameserver 120 can find the preferences or subscriber information used by options 27").

Thus, the purpose Treuhaft's control information is to identify the owner or subscriber associated with the host device 105 seeking access to the Internet. Ex. 1003, ¶ 168. Accordingly, a POSA would have understood that any piece of information associated with the user or subscriber would be used as the control information in the DNS query, such as an IP address or hierarchical domain name. Ex. 1003, ¶ 168. Indeed, according to its dictionary definition, a hierarchical domain name serves the exact purpose of Treuhaft's user identifier, subscriber identifier, device identifier, or other control information: "An address of a network connection <u>that identifies the owner</u> of that address in a hierarchical format." Ex. 1012, 4. And hierarchical domain names were routinely used for this purpose long before the '040 Patent. Ex. 1003, ¶ 168. Accordingly, a POSA would have found it obvious to use a hierarchical domain name for the host device 105 as an alternative, or in addition to, Treuhaft's user identifier, subscriber identifier, device identifier, or other type of control information. Ex. 1003, ¶ 168.

Accordingly, Treuhaft discloses, or at least suggests, *"receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device ...* ," as claimed.

c. [1.2] *"resolving the received server domain name to obtain a service server Internet protocol (IP) address; and"*

Treuhaft discloses, or at least suggests, this element. Ex. 1003, ¶¶ 170-171. Specifically, Treuhaft discloses resolving the received server domain name (domain name associated with URL contained in the modified DNS query) to obtain a service server Internet protocol (IP) address. *Id.* Treuhaft explains, "in step 530, the DNS query is received. In step 535, the DNS query is parsed or otherwise processed at the DNS nameserver to determine the domain name and the control information." Ex. 1008, ¶ [0063]. Then, "[i]n step 540, <u>an IP address is determined</u> using one or more DNS resolution options or features and the control information. In one example, the domain name is resolved to its corresponding IP address." *Id.*, ¶ [0064] (emphasis added).

Accordingly, Treuhaft discloses, or at least suggests, "*resolving the received server domain name to obtain a service server Internet protocol (IP) address*," as claimed.

d. [1.3] "*discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*"

Treuhaft discloses, or at least suggests, this element. Ex. 1003, ¶ 172-182. As explained below, the DNS name server 120 of Treuhaft maintains subscriber information 208 for various users or subscribers, which corresponds to the claimed "preset list." *See* Ex. 1008, ¶¶ [0028], [0029], [0034], [0036], [0039], [0054], [0060], [0064], FIG. 2 (subscriber information 280); Ex. 1003, ¶ 172. If the subscriber information 208 for the user or subscriber indicates to block access to the IP address resolved from the modified DNS query—such as an IP address for an inappropriate website—the DNS name server 120 of Treuhaft discards the modified DNS query by not returning that IP address to the host device 105. *See, e.g.*, Ex. 1008, ¶¶ [0027], [0028]; Ex. 1003,¶ 172.

Specifically, Figure 2 of Treuhaft shows the DNS name server 120 storing subscriber information 280 for the users or subscribers of the system in memory 220:

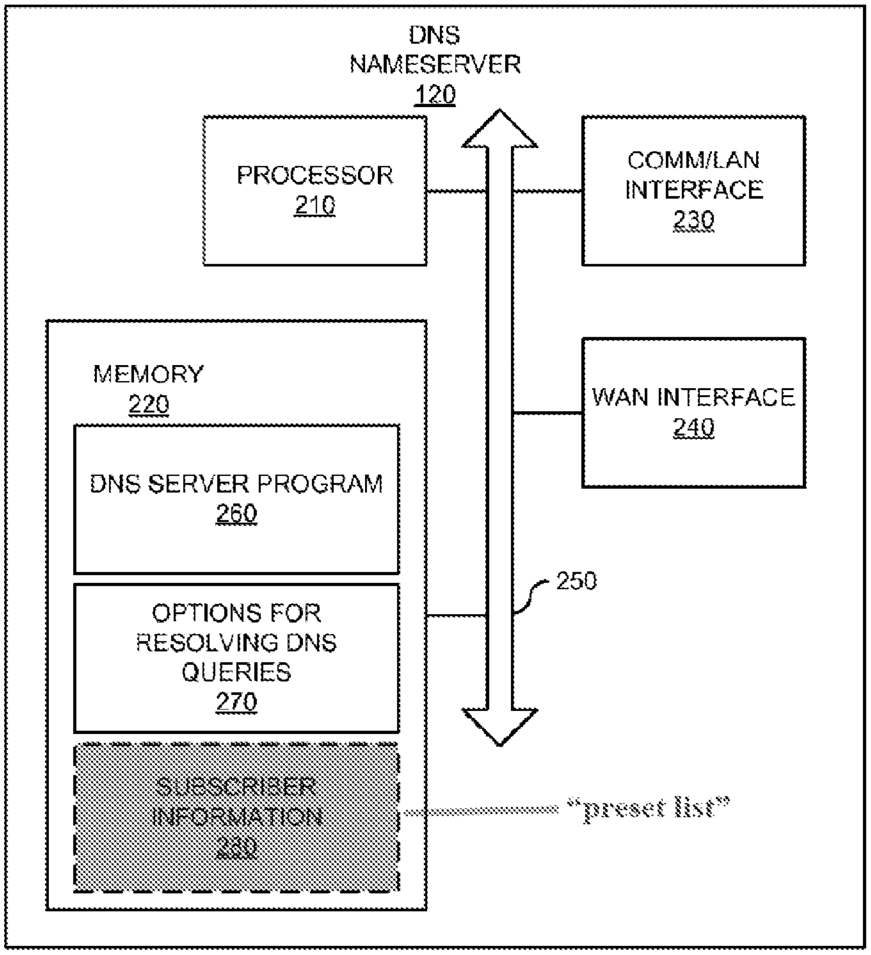


FIG. 2

Treuhaft, FIG. 2*

As highlighted above, the subscriber information 280 corresponds to the claimed preset list. Ex. 1003, ¶ 173. "The subscriber information can include preferences or other settings for how a user or subscriber wishes to control domain name resolution within the DNS resolution features." Ex. 1008, ¶ [0060]. "For example, a user or subscriber may establish subscriber information that instructs DNS nameserver 120 to alter responses to DNS requests that are associated with adult web sites, potential phishing or pharming sites, and other sites deemed inappropriate by the user or containing material illegal in the country of the user." *Id.*, ¶ [0028]. The "subscriber information associated with the user may be used to alter the IP address in a DNS response that the user receives." *Id.*

In steps 535-550 of Figure 5, the name server 120 of Treuhaft applies the subscriber information for the user or subscriber to the modified DNS query in determining how to respond to the modified DNS query. *See id.*, ¶¶ [0064]-[0066]. This process corresponds to the claimed "discarding the service request packet":

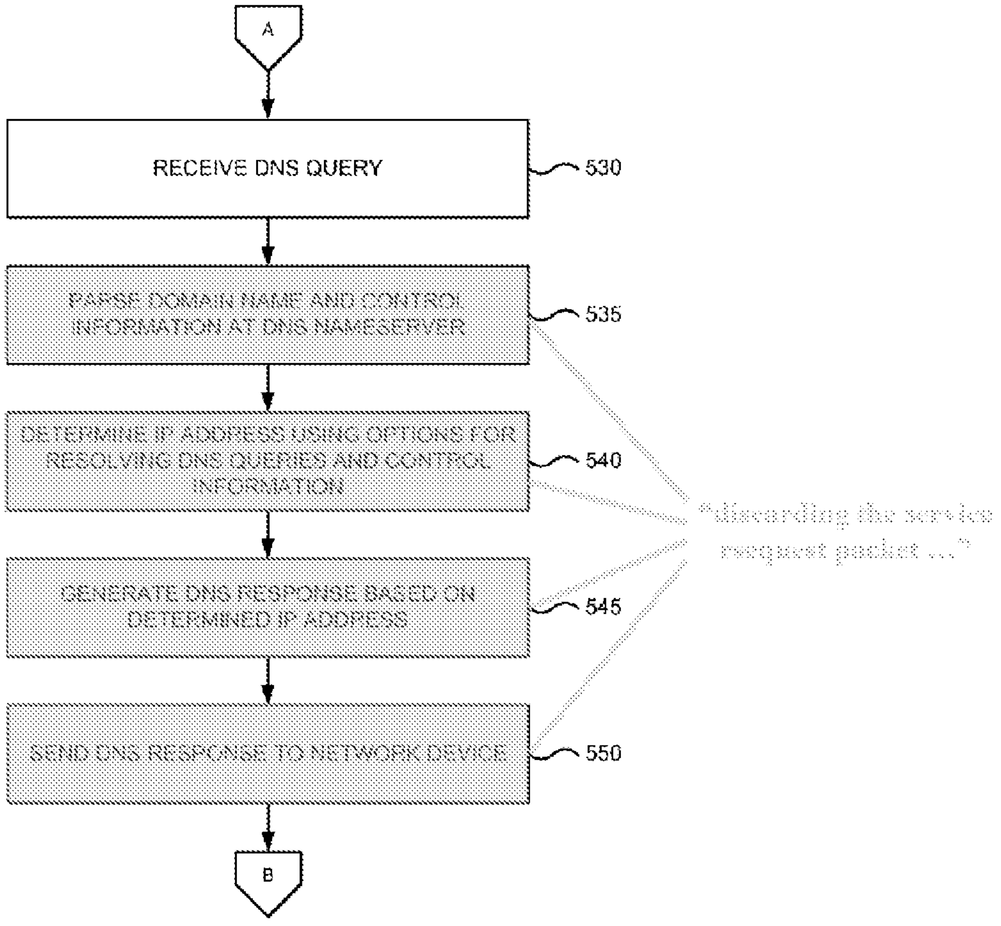

A
RECEIVE DNS QUERY
530
PARSE DOMAIN NAME AND CONTROL INFORMATION AT DNS NAMESERVER
535
DETERMINE IP ADDRESS USING OPTIONS FOR RESOLVING DNS QUERIES AND CONTROL INFORMATION
540
GENERATE DNS RESPONSE BASED ON DETERMINED IP ADDRESS
545
SEND DNS RESPONSE TO NETWORK DEVICE
550
"discarding the service request packet ..."
B

FIG. 5B

Treuhaft, FIG. 5B*

Specifically, after receiving the modified DNS query, the DNS name server 120 parses the control information in the modified DNS query to identify the particular user or subscriber that sent the DNS query, and retrieves that user or subscriber's subscriber information 280 in step 535. *Id.*, ¶ [0064]. Then, in step 540, using the user or subscriber's subscription information 280, the DNS name server 120 "make[s] a decision whether to use the corresponding IP address or another IP address when generating a DNS response based on applying one or more DNS resolution options or features". *Id.* ¶ [0065]; Ex. 1003, ¶ 175. For example, rather than return the <u>resolved</u> IP address requested by the host device 105, "DNS nameserver 120 may determine to substitute the IP address of a website that provides information why the domain name is being block[ed], forwarded, filtered, or otherwise includes material the user has expressed a desire to control." Ex. 1008, ¶ [0065] (emphasis added); Ex. 1003, ¶ 175. Then, in steps 545 and 550, the DNS name server 120 respectively generates a DNS response "substitut[ing] [the] IP address based on applying one or more of the available DNS resolution options, filters, or features" and sends the DNS response with the substituted IP address to the host device 105. Ex. 1008, ¶ [0066]; Ex. 1003, ¶ 175.

As explained above, the broadest reasonable interpretation of "discarding the service request packet" includes preventing unauthorized access to the resolved service server IP address. *Supra* Section I.B.1. Treuhaft's process in steps 535-550 of Figure 5B prevents the host device 105 from unauthorized access to the resolved IP address. Ex. 1003, ¶ 176. This is because the DNS name server 120 alters or substitutes the resolved IP address for a different IP address in the DNS response if the subscriber information deems access to that IP address unauthorized. Ex. 1008, ¶¶ [0065], [0066]; *see also id.*, ¶¶ [0028], [0035]; Ex. 1003, ¶ 176. For example, Treuhaft may instead "substitute the IP address of a website that provides information why the domain name is being block[ed], forwarded, filtered, or otherwise includes material the user has expressed a desire to control." Ex. 1008, ¶ [0065]; *see also id.*, ¶ [0032] (the DNS name server 120 "respond[s] with another IP address that, for example, redirects the user to a website with additional information for the reason why the corresponding IP address was not returned"). By responding to the DNS query with a different IP address than the one requested and blocking the requested IP address, Treuhaft prevents unauthorized access to the resolved IP address. Ex. 1003, ¶ 176. Thus, Treuhaft discloses or suggests "discarding the service request packet," as claimed. *Id.*

Moreover, even if the claimed discarding were construed narrowly to mean providing no DNS response at all to the DNS query, this option is suggested based on Treuhaft's disclosure of "blocking" or "not returning" the IP address for an unauthorized site. *See* Ex. 1008, ¶¶ [0032], [0065]; Ex. 1003, ¶ 177

Treuhaft's subscriber information discloses, or at least suggests, a "preset list" containing "a preset service server IP address corresponding to the received terminal domain name," as claimed. Ex. 1003, ¶ 178. In Treuhaft, the DNS name server 120 applies the subscriber information to "make a decision whether to use the corresponding [resolved] IP address or another IP address" if the resolved IP address is not authorized. Ex. 1008, ¶ [0065]; *see also id.* ¶¶ [0065], [0066] (if the resolved IP address is not authorized, the name server 120 returns a "substitute IP address"). Treuhaft gives an example in which, applying the subscriber information, the "DNS nameserver 120 may respond with the [resolved] IP address of 'www.cnet.com' …. or may respond with another IP address that, for example, redirects the user to a website with additional information for the reason why the corresponding IP address was not returned." *Id.*, ¶ [0032].

Based on this disclosure, a POSA would have understood that the DNS name server 120 of Treuhaft checks the resolved IP address for "www.cnet.com" against known authorized/unauthorized IP addresses in deciding whether to return the resolved IP address for "www.cnet.com" or another IP address. Ex. 1003, ¶ 179. That is, if the resolved IP address is authorized for the user/subscriber of the host device 105—i.e., "belongs to a preset service server IP address corresponding to the received terminal domain name," as claimed—the DNS name server 120 returns the resolved IP address. *Id.* Otherwise, the DNS name server 120 discards the DNS query by returning a different IP address, thus preventing access to the resolved IP address as discussed above. *Id.*

Given Treuhaft's IP address comparison, Treuhaft discloses or at least suggests that the subscriber information includes a "preset list" of one or more authorized or unauthorized IP addresses, as claimed. *Id.*, ¶ 180. Indeed, in order to make this IP address comparison, Treuhaft necessarily must store a list of authorized/unauthorized IP addresses somewhere, and the subscriber information 208 for the particular user/subscriber is the most logical place. As explained above, the DNS name server 120 stores the subscriber information 280 in memory 220. Ex. 1008, ¶ [0034], FIG. 2. It would have been obvious to implement Treuhaft's subscriber information as a preset list of authorized/unauthorized IP addresses. Ex. 1003, ¶ 180. Moreover, it would have been obvious to store this list of authorized/unauthorized IP addresses in the memory 220 as part of the subscriber information 280 for the particular user/subscriber. Ex. 1003, ¶ 180.

If it is argued that Treuhaft's "preset list" functions as a blacklist for discarding the DNS query if the resolved IP address belongs to the list, rather than "does not belong" (emphasis added) as claimed, it would have been obvious to alternatively or additionally implement Treuhaft's list as a whitelist. *Id.*, ¶ 181. Long before the claimed priority date of the '040 patent, blacklists and whitelists were known and used interchangeably and/or together in the same system to control access to sites. *Id.* Depending on the knowledge of the administrator configuring the access controls for users or subscribers of the system, it would have been obvious to use a whitelist of authorized IP addresses, a blacklist of unauthorized IP addresses, or a combination of both a whitelist and a blacklist. *Id.* For example, if the administrator wanted to configure Treuhaft's system so that a user or subscriber may only access certain known IP addresses, the administrator would have found it obvious to use a whitelist of authorized IP addresses. But if the administrator wanted to allow access to all sites except those specifically deemed inappropriate, for example, the administrator would have found it obvious to use a blacklist. *Id.*

Accordingly, for at least the reasons above, Treuhaft discloses or suggests "*discarding the service request packet if the resolved service server IP address does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list*," as claimed.

e. [1.4] "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device.*"

Treuhaft discloses, or at least suggests, this element. Ex. 1003, ¶¶ 183-185. As explained above, the subscriber information 280 of Treuhaft discloses or suggests a preset list of authorized/unauthorized IP addresses that a given user or subscriber has or does not have authorization to access. *Supra* Section II.B.1.d. Moreover, Treuhaft's list of authorized/unauthorized IP addresses for users or subscribers discloses or suggests the claimed "corresponding[] … plurality of accessible service server IP addresses under an access authority of the terminal device," as recited in element [1.4]. Ex. 1003, ¶ 183.

Treuhaft explains that its system allows "DNS resolution [to] be controlled on a per-request basis for <u>each individual user or device</u>." Ex. 1008, ¶¶ [0008], [0024] (emphasis added). This suggests the DNS name server 120 contains subscriber information 280 for multiple users or subscribers, as the DNS name server 120 would need this information to process DNS queries on a user-by-user or subscriber-by-subscriber basis. Ex. 1003, ¶ 184. As discussed above, the subscriber information 280 includes, or suggests, a list of authorized/unauthorized IP addresses for its users or subscribers. *Id.* Thus, it would have been obvious to arrange Treuhaft's list such that the user identifier, subscriber identifier, device ID, CLIENTID, or hierarchical domain name (claimed terminal domain name) associated with each host device 105 (claimed terminal device) maps to the corresponding authorized/unauthorized IP address(es) for that user or subscriber. *Id.* Mapping the host device identifiers to their corresponding authorized/unauthorized IP addresses in this manner would allow the DNS name server 120 to check the resolved IP address against the authorized/unauthorized IP addresses for that user or subscriber when processing a DNS query from that user or subscriber. *Id.*

Accordingly, Treuhaft discloses, or at least suggests, "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with a plurality of accessible service server IP addresses under an access authority of the terminal device*," as claimed.

2. Dependent Claim 4

a. [4.1] "*The method according to claim 1, wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:*

***if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device.*"**

Claim 4 depends from independent claim 1 and recites the additional elements quoted above. Treuhaft anticipates and renders obvious independent claim 1 for the reasons discussed above. *Supra* Section II.B.1. Moreover, Treuhaft discloses, or at least suggests, the additional elements of claim 4. Ex. 1003, ¶¶ 186-191.

As explained above, Treuhaft discloses or suggests that the DNS name server 120 uses a list of authorized/unauthorized IP addresses for each user or subscriber of the system to determine whether the resolved IP address is authorized/unauthorized. *Supra* Section; Ex. 1003 ¶ 187. If the resolved IP address authorized, the "DNS nameserver 120 … use[s] the corresponding IP address of the domain name"—i.e., the resolved IP address—in the DNS response to the host device 105 in steps 545 and 550. Ex. 1008, ¶ [0066]. As shown in Figure 5C of Treuhaft, the host device 105 subsequently receives the DNS response from the name server 120 (step 555), determines the resolved IP address from the DNS response (step 560), and forwards the IP address to an application running on the host device 105 (step 565). *See* Ex. 1008, ¶ [0067], FIG. 5C.

A POSA would have understood that, upon forwarding the IP address to the application on the host device 105, the application connects to the server associated with that IP address. Ex. 1003, ¶ 188. Indeed, Treuhaft teaches that the application running on the host device, which sent the DNS query for the IP address, is a web browser. *See* Ex. 1008,¶¶ [0025], [0057]. And a web browser is an application, used to browse the Internet, that sends a DNS query for an IP address to a DNS server, receives a DNS response containing the IP address from the DNS server, and connects to a server located at the IP address. Ex. 1003, ¶ 188. If it is argued that Treuhaft does not expressly disclose connecting to the server using the IP address, a POSA would have found it obvious to have the browser application connect to the server located at the returned IP address because Treuhaft seeks to enable Internet connections with DNS-based access controls, and the purpose of a browser is to access the Internet using IP addresses returned from DNS servers. *See, e.g.*, Ex. 1008, ¶¶ [0022]-[0024]; Ex. 1003, ¶ 188.

Accordingly, any combination of steps 545, 550 (Figure 5B, reproduced below) performed by the name server 120 and steps 555-565 (Figure 5C) performed by the host device 105 discloses or suggests "if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection … ," as recited in claim 4. Ex. 1003, ¶ 189. As highlighted below:

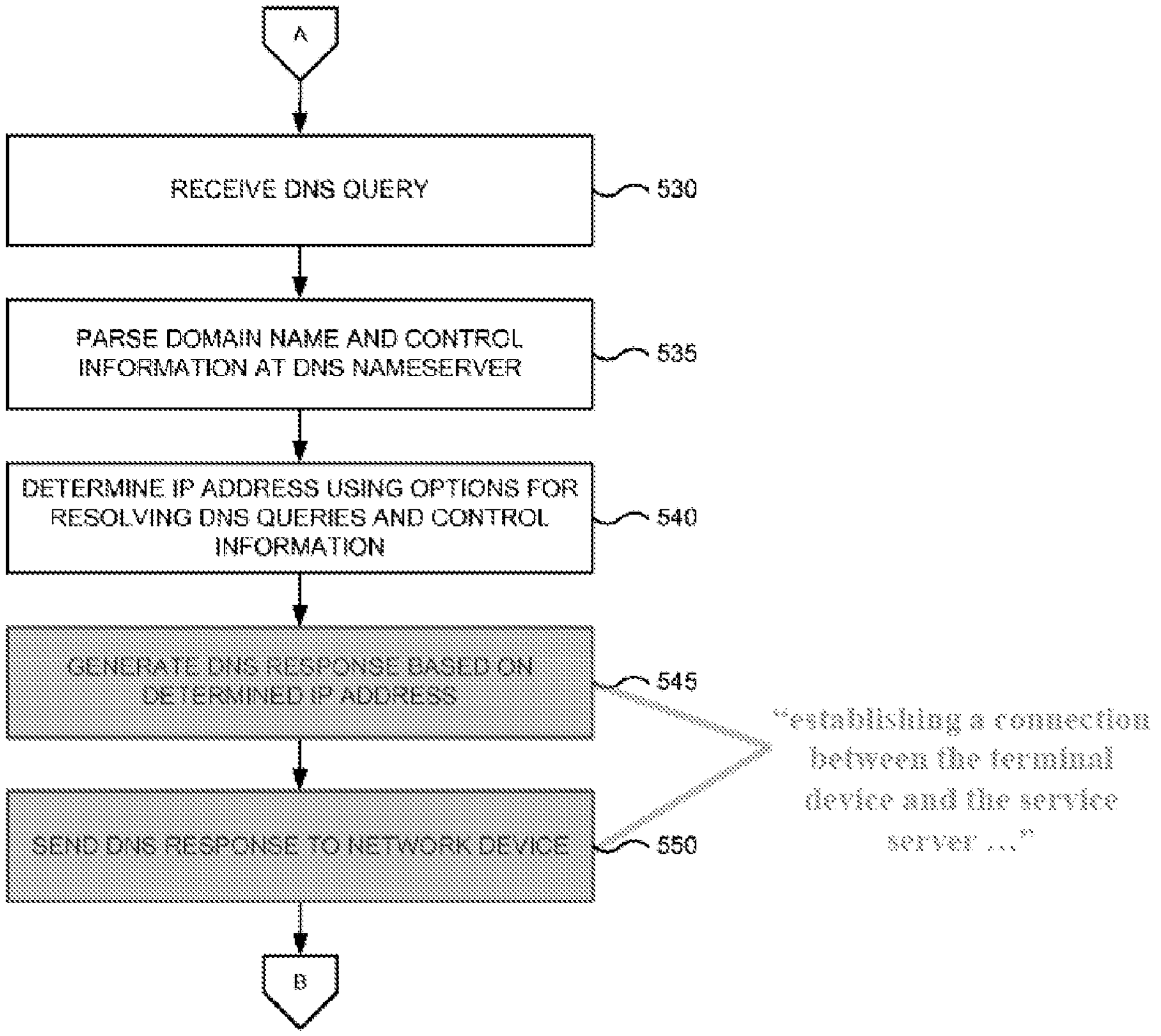

A
RECEIVE DNS QUERY
530
PARSE DOMAIN NAME AND CONTROL INFORMATION AT DNS NAMESERVER
535
DETERMINE IP ADDRESS USING OPTIONS FOR RESOLVING DNS QUERIES AND CONTROL INFORMATION
540
GENERATE DNS RESPONSE BASED ON DETERMINED IP ADDRESS
545
SEND DNS RESPONSE TO NETWORK DEVICE
550
"establishing a connection between the terminal device and the service server …"
B

FIG. 5B

Treuhaft, FIG. 5B*

It is noted that, under the broadest reasonable interpretation standard, the DNS name server 120's sending the DNS response with the resolved IP address to the host device 105, even taken alone, discloses or suggests "establishing a connection," as claimed. Ex. 1003, ¶ 190. This is because, as explained above, the application running on the host device 105 connects to the intended server using the IP address contained in the DNS response upon receiving the DNS response. *See, e.g.*, Ex. 1008, ¶ [0067], FIG. 5C; *see also id.*, ¶¶ [0025], [0057]; Ex. 1003, ¶ 190. That is, the process of returning the DNS response to the host device initiates the establishment of the connection between the host device 104 (claimed terminal device) and the server located at the IP address (claimed service server corresponding to the service server IP address). Ex. 1003, ¶ 190.

Accordingly, Treuhaft discloses or suggests "*wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises: if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device*," as recited in claim 4.

3. Dependent Claim 5

a. [5.1] "*The method according to claim 1, wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:*

***if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device.*"**

Claim 5 depends from independent claim 1 and recites the additional elements quoted above. Treuhaft anticipates and renders obvious independent claim 1 for the reasons discussed above. *Supra* Section II.B.1. Moreover, Treuhaft discloses, or at least suggests, the additional elements of claim 5 because, as explained below, the DNS name server 120 uses a resource record contained in the DNS query to determine the DNS extension capabilities of the host device 105 (claiming determining a service type). Ex. 1003, ¶¶ 192-196.

As Treuhaft explains, the DNS query 400 contains fields the host device 105 uses "to advertise its own extended capabilities to the message receiver (e.g., DNS nameserver 120)." *See* Ex. 1008, ¶¶ [0040]-[0041], FIG. 4. "This may be accomplished through the inclusion of an OPT pseudo-RR in the additional data section of a request or response. The OPT pseudo-RR may include one or more EDNS options." *Id.*, ¶ [0041]. OPT pseudo-RR refers to an options resource record contained in a DNS query sent according to the Extension Mechanisms for DNS (EDNS) specification. Ex. 1003,¶ 193. A device uses this resource record in a DNS query to identify DNS extension capabilities to the DNS server. *Id.* For example, one such extension mechanism is DNS Security Extensions (DNSSEC), securing data exchanged over DNS using cryptography. *Id.*

A POSA would have understood that the host device 105 in Treuhaft uses the options resource record to indicate to the DNS name server 120 that it supports DNSSEC or another type of DNS extension (claimed service type). *Id.*, ¶ 194; *see also* Ex. 1008, ¶ [0041]. In the case of a DNSSEC-enabled host device 150, for example, the host device 105 includes information in the options resource record of the DNS query indicating that it supports DNSSEC. Ex. 1003, ¶ 194. Upon receiving the DNS query, the DNS name server 120 unpacks the DNS query, and determines from this resource record that the host device 105 supports DNSSEC. *Id.* Accordingly, the DNS server responds to the host device 105 according to the DNSSEC protocol, such as by engaging in a cryptographic handshake routine with the host device 105 and/or encrypting its DNS response. *Id.* Thus, according to EDNS protocol, the DNS name server 120 of Treuhaft "determin[es] a service type of the service request," as claimed, based on information contained in the options resource record. Ex. 1003, ¶ 194; *see also* Ex. 1008, ¶¶ [0040]-[0041].

Treuhaft further discloses that the DNS name server 120 determines the service type of the DNS query "according to the terminal domain name of the terminal device," as claimed. Ex. 1003, ¶ 195. For example, Treuhaft explains that the host device 105 may include its CLIENTID in the options resource record to identify its DNS extension capabilities to the DNS name server 120:

> In some embodiments, host device 105 can define a new EDNS option called CLIENTID for control of user, device, or vendor-specific DNS server behavior. <u>The CLIENTID option may appear in an OPT pseudo-RR in the additional data section of a request</u>. In general, a CLIENTID option applies to the DNS request that it accompanies. Thus, the CLIENTID can allow a per-request control of each DNS message.

Ex. 1008, ¶ [0042] (emphasis added). Thus, the DNS name server 120 of Treuhaft uses the CLIENTID in the options resource record of the DNS query to determine the host device 105's (claimed terminal device) DNS extension capabilities (claimed service type). *Id.*; *see also* Ex. 1003, ¶ 195. As explained above, the CLIENTID corresponds to the claimed terminal domain name because it identifies the user or subscriber associated with the host device 105. *Supra* Section II.B.1.b; *see also* Ex. 1008, ¶¶ [0042]-[0045]; Ex. 1003, ¶ 195. And because the DNS name server 120 uses the CLIENTID (claimed terminal domain name) in the options resource record of the DNS query to determine the host device 105's DNS extension capabilities, Treuhaft discloses or suggests determining the service type of the DNS query "according to the terminal domain name of the terminal device," as claimed. Ex. 1003, ¶ 195.

For at least the reasons above, Treuhaft discloses or suggests "*wherein, after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises: if the resolved service server IP address belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device*," as recited in claim 5.

4. Independent Claim 6

As set forth below, Treuhaft discloses or suggests of the elements of independent claim 6 for reason similar to those discussed above for independent claim 1.

a. [6.pre] "*A deep packet inspection (DPI) device comprising a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising:*"

To the extent the preamble is limiting, Treuhaft discloses this element. Ex. 1003, ¶ 198. Specifically, the DNS name server 120 of Treuhaft corresponds to the claimed DPI device:

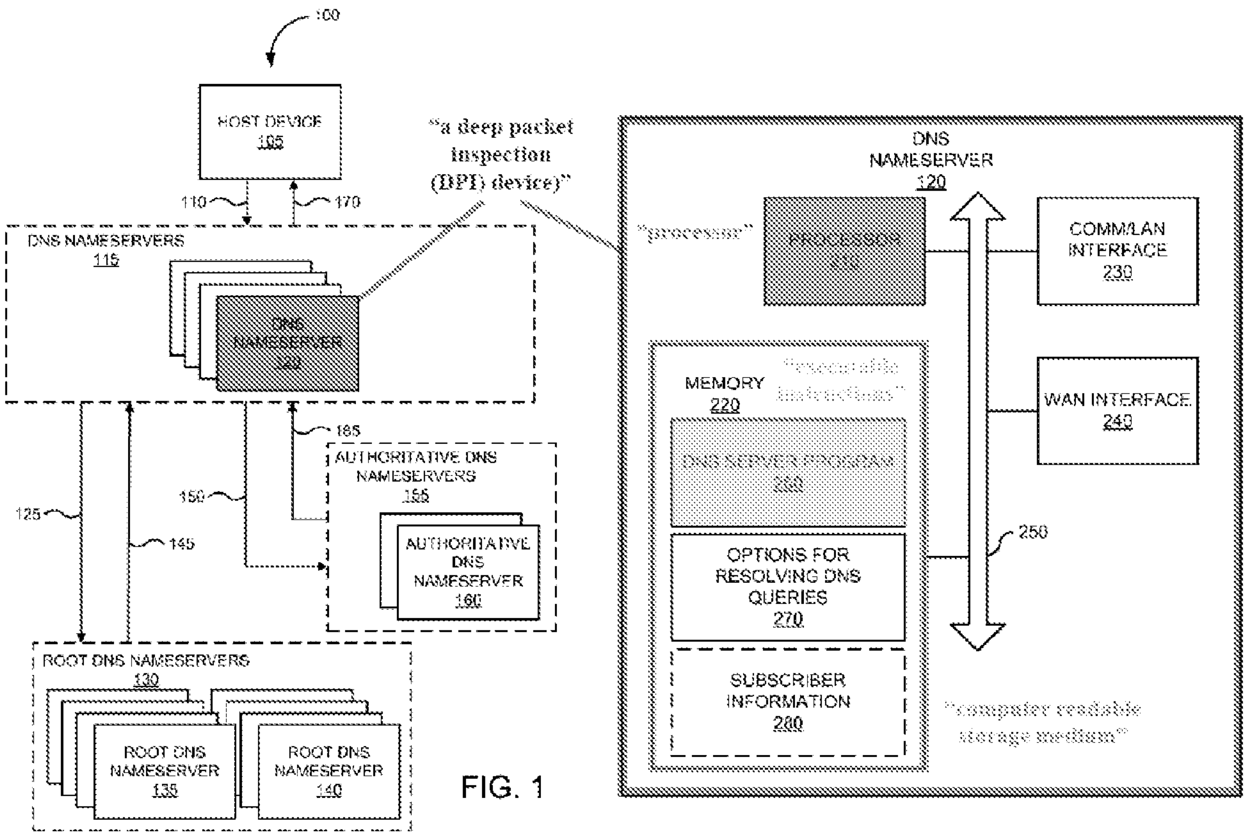


FIG. 2

Treuhaft, FIGS. 1, 2*

As highlighted above, the DNS name server 120 has a processor 210 (claimed processor) and a memory 200 (claimed computer readable storage medium) storing a DNS server program 260 (claimed executable instructions), executed by the processor 210 to perform the functions of the DNS name server 120. Ex. 1008, ¶¶ [0033]-[0034], [0038]; Ex. 1003. ¶ 198.

Accordingly, Treuhaft discloses or suggests *"[a] deep packet inspection (DPI) device comprising a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method,"* as claimed.

b. [6.1] "*receiving a service request packet sent by a terminal device, wherein the service request packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request packet sent by the terminal device;*"

See the discussion above for element [1.1]. *Supra* Section II.B.1.b.

c. [6.2] "*resolving the server domain name to obtain a service server Internet protocol (IP) address; and*"

See the discussion above for element [1.2]. *Supra* Section II.B.1.c.

d. [6.3] "*discarding the packet if the service server IP address resolved does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*"

See the discussion above for element [1.3]. *Supra* Section II.B.1.d.

e. [6.4] "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with accessible service server IP addresses under an access authority of the terminal device.*"

See the discussion above for element [1.4]. *Supra* Section II.B.1.e.

5. Dependent Claim 9

a. [9.1] "*The DPI device according to claim 6, wherein after the resolving the received server domain name to obtain the service*

server Internet protocol (IP) address, the method further comprises:

if the service server IP address resolved belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, establishing a connection between the terminal device and the service server corresponding to the service server IP address, to enable the service server to provide a service corresponding to the service request of the terminal device to the terminal device."

Claim 9 depends from independent claim 6 and recites the additional elements quoted above. Treuhaft anticipates and renders obvious independent claim 6 for the reasons discussed above. *Supra* Section II.B.4. Moreover, claim 9 recites subject matter virtually identical to that discussed above for dependent claim 4. Accordingly, for the reasons discussed above in connection with claim 4, Treuhaft discloses, or at least suggests, the subject matter of claim 9. *Supra* Section II.B.2.

6. Dependent Claim 10

a. **[10.1]** "***The DPI device according to claim 6, wherein after the resolving the received server domain name to obtain the service server Internet protocol (IP) address, the method further comprises:***

if the service server IP address resolved belongs to the preset service server IP address corresponding to the received terminal domain name in the preset list, determining a service type of the service request according to the terminal domain name of the terminal device."

Claim 10 depends from independent claim 6 and recites the additional elements quoted above. Treuhaft anticipates and renders obvious independent claim 6 for the reasons discussed above. *Supra* Section II.B.4. Moreover, claim 10 recites subject matter virtually identical to that discussed above for dependent claim 5. Accordingly, for the reasons discussed above in connection with claim 5, Treuhaft discloses, or at least suggests, the subject matter of claim 10. *Supra* Section II.B.3.

7. Independent Claim 11

As set forth below, Treuhaft discloses or suggests of the elements of independent claim 11 for reason similar to those discussed above for independent claims 1 and 11.

a. [11.pre] "*A system, comprising: a deep packet inspection (DPI) device; and a terminal device:*"

To the extent the preamble is limiting, Treuhaft discloses this element. Ex. 1003, ¶ 270. As highlighted in Figure 1 below, Treuhaft discloses a DNS system 100 (claimed system) comprising a DNS name server 120 (claimed DPI device) and a host device 105 (claimed terminal device):

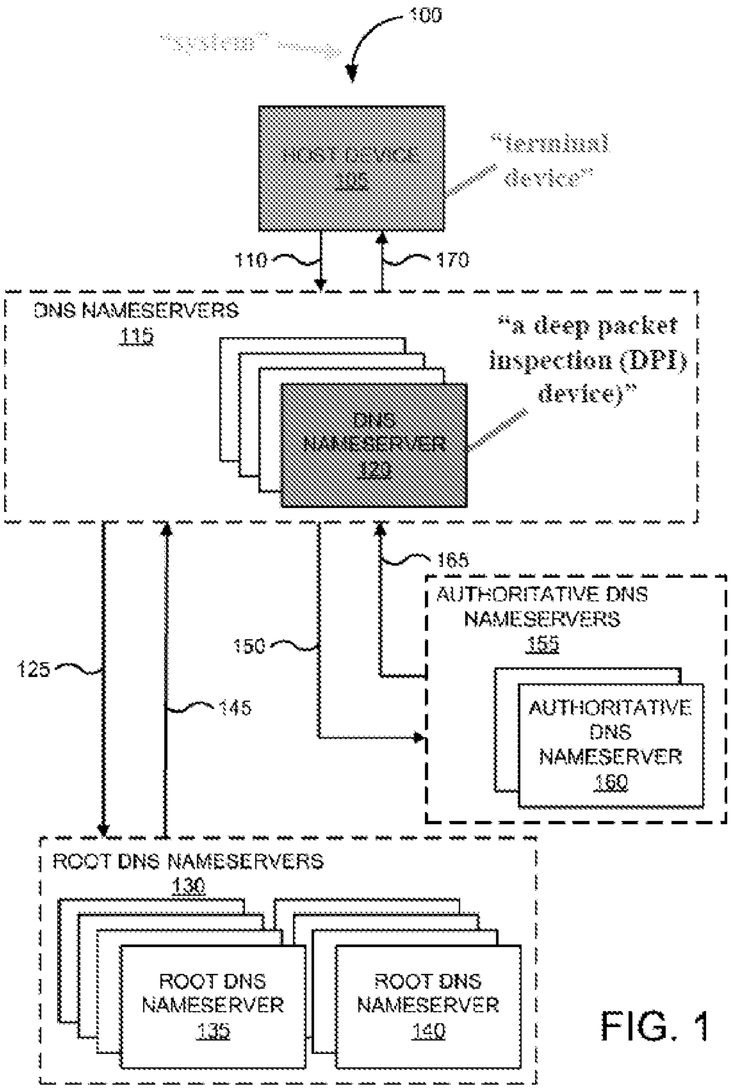


FIG. 1

Treuhaft, FIG. 1*

Accordingly, Treuhaft discloses or suggests *"[a] system, comprising: a deep packet inspection (DPI) device; and a terminal device,"* as claimed.

b. [11.1] [*a terminal device*] "*configured to send a service request packet to the DPI device, wherein the packet carries a terminal*

***domain name indicating the terminal device and a server domain name indicating a service server required by the service request sent by the terminal device;*" and**

***"the DPI device having a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising: receiving the service request packet sent by the terminal device;*"**

As explained above for element [1.1], in Treuhaft, the host device 105 (claimed terminal device) is configured to send a DNS query (claimed service request packet) to the DNS name server 120 (claimed DPI device). *Supra* Section II.B.1.b. Additionally, the DNS query (claimed service request packet) carries control information (claimed terminal domain name) indicating the host device 105 (claimed terminal device) and a domain name of a URL (claimed server domain name) indicating a server to which the host device 105 seeks to connect (claimed service server required by the service request sent by the terminal device). *Id.*

As explained above for element [6.pre], the DNS name server 120 (claimed DPI device) has a processor 210 (claimed processor) and a memory 200 (claimed computer readable storage medium) storing a DNS server program 260 (claimed executable instructions), executed by the processor 210 to perform the functions of the DNS name server 120. Ex. 1008, ¶¶ [0033]-[0034], [0038]; Ex. 1003 ¶ 210. And, part of that method includes receiving the DNS query (claimed service request packet) sent by the host device 105 (claimed terminal device). *Supra* Section II.B.1.b; *see also* Ex. 1008, ¶¶ [0063] ("In step 525, the modified DNS query is sent to a DNS nameserver. For example, the modified DNS query may be sent to DNS nameserver 120."), ¶ [0064] ("in step 530, the DNS query is received"), FIG. 5B (step 550), FIG. 5C (step 555).

Accordingly, Treuhaft discloses or suggests "*a terminal device ... configured to send a service request packet to the DPI device, wherein the packet carries a terminal domain name indicating the terminal device and a server domain name indicating a service server required by the service request sent by the terminal device*" and "*the DPI device having a hardware processor and a non-transitory computer readable storage medium including executable instructions that, when executed by the processor perform a method comprising: receiving the service request packet sent by the terminal device*," as claimed.

c. [11.2] "*resolving the server domain name to obtain a service server*

***Internet protocol (IP) address; and*"**

See the discussion above for element [1.2]. *Supra* Section II.B.1.c.

d. [11.3] "*discarding the packet if the service server IP address resolved does not belong to a preset service server IP address corresponding to the received terminal domain name in a preset list,*"

See the discussion above for element [1.3]. *Supra* Section II.B.1.d.

e. [11.4] "*wherein in the preset list the terminal domain name of each terminal device is correspondingly provided with accessible service server IP addresses under an access authority of the terminal device.*"

See the discussion above for element [6.4]. *Supra* Section II.B.1.e.

C. Ground 4: Treuhaft in View Sorenson Renders Obvious Claims 1, 4-6, and 9-11 of the '040 Patent Under § 103

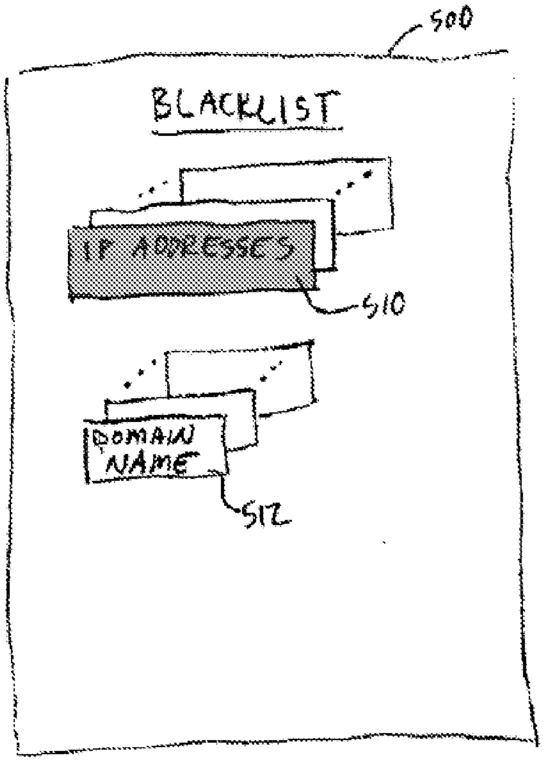
Patent Owner may potentially contend Treuhaft's subscriber information 280 constitutes a preset list of authorized/unauthorized server domain names, rather than IP addresses. On this basis, Patent Owner may contend Treuhaft discloses determining whether the server domain name is on the preset list—not the IP address resolved from that domain name—and so Treuhaft does not disclose "discarding the service request packet if the [resolved] service server IP address does not belong to a preset service server IP address … in a preset list," as recited in elements [1.3], [6.3], and [11.3]. Patent Owner would be incorrect at least because, as discussed above, Treuhaft at least suggests that the subscriber information 280 contains a list of authorized and/or unauthorized IP addresses for each user or subscriber. *Supra* Sections II.B.1.d, e. In the case of an authorized IP address list, a POSA would have understood Treuhaft to disclose discarding the DNS query if the resolved IP address is not on the authorized list. *Id.* And, moreover, a POSA would have found it obvious to use a list of authorized IP addresses, unauthorized IP addresses, or both for the purpose of controlling access to the Internet. *Id.*

Even if Patent Owner were correct, however, the additional disclosure of Sorenson renders obvious elements [1.3], [6.3], and [11.3]. Ex. 1003, ¶¶ 215-231. Accordingly, as explained below, the combination of Treuhaft in view of Sorenson further renders obvious claims 1, 4-6, and 9-11 of the '040 Patent.

1. **Sorenson Discloses Discarding the Service Request Packet "<u>if the [Resolved] Service Server IP Address</u> Does Not Belong to a Preset Service Server IP Address ... in a Preset List" (Elements [1.3], [6.3], and [11.3])**

Sorenson discloses or suggests elements [1.3], [6.3], and [11.3] by denying a connection if the IP address resolved from a domain name in the connection request is found on an IP address blacklist. Ex. 1003, ¶ 217.

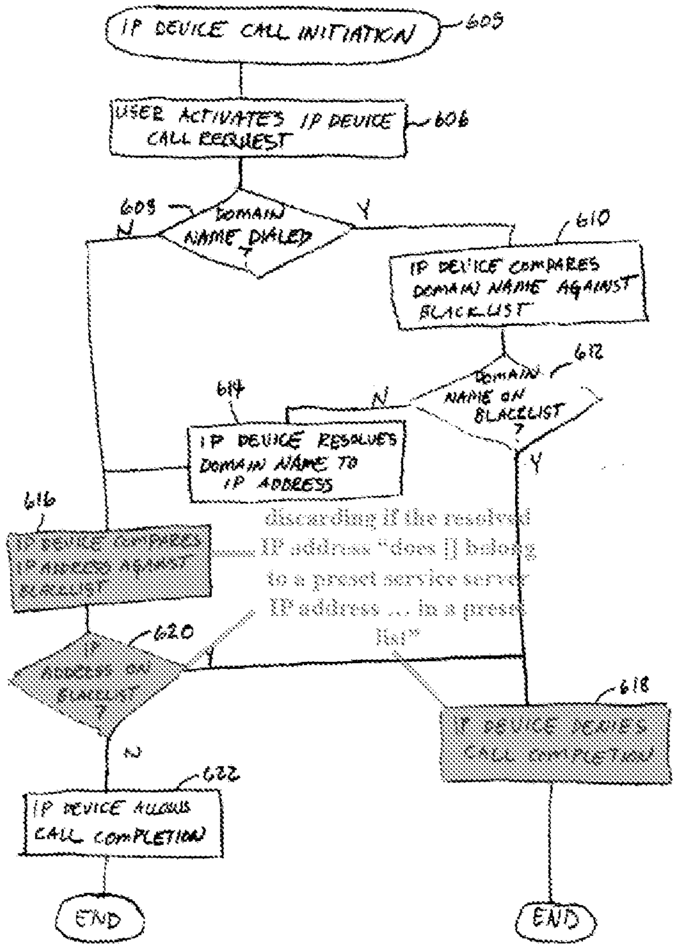
Specifically, Sorenson discloses a "system and method for blocking access by a network device to specific network resources by comparing a specific resource identifier against entries in a blacklist and facilitating a connection accordingly." Ex. 1009, Abstract. In Sorenson, the system receives "a call request for the establishment of a communication session between IP device 12 and associated service 20. *Id.*, ¶¶ [0027]; *see also id.*, ¶¶ [0031], [0032]. The IP device 12, call request, and service 20 of Sorenson respectively correspond to the claimed terminal device, service request packet, and service server. Ex. 1003, ¶ 218. Like the service request packet of the '040 Patent, Sorenson's call request "include[s] a specific identifier such as an entered IP address, domain name, or conventional phone number or name resolved into one of an IP address or domain name," which corresponds to the claimed server domain name. Ex. 1009, ¶ [0031]; Ex. 1003, ¶ 218.



Before granting the call request, the system of Sorenson performs a two-stage blacklist check to determine whether to establish the connection or discard the request. *See* Ex. 1009, ¶¶ [0031]-[0032], FIG. 6; Ex. 1003, ¶ 219. As shown in Figure 2 of Sorenson above, the blacklist 500 contains both blacklisted domain name names 512 and <u>blacklisted IP addresses 510</u>. *See* Ex. 1009, ¶ [0028], FIG. 2. The blacklist 500 of blacklisted IP addresses 510 corresponds to the claimed preset list. Ex. 1003, ¶ 219.

First, as shown in step 610 of Figure 6, Sorenson performs a domain-name-blacklist check by "compar[ing] 610 the domain name against the blacklist 500′ (FIG. 2) to determine 612 if the domain name is located within the blacklist 500′." Ex. 1009, ¶ [0031]. "If the domain name utilized for initiating the call is located with the blacklist 500′, then the IP device denies 618 the completion of the call and may alternatively notify the user of such denial." *Id.* But "[i]f the domain name is not on the blacklist, then" Sorenson performs an IP-address-blacklist check before establishing the connection. *Id.* Specifically, Sorenson "resolves … the domain name into an IP address for further comparison" in step 614, *id.*, and then "compares … the IP address against the blacklist 500'" in step 616, *id.*, ¶ [0032]. If the IP address is located within the blacklist 500', Sorenson denies the connection. *Id.*, ¶ [0032], FIG. 6 (step 618). But if the IP address is not found on the blacklist 500', Sorenson establishes the connection. *Id.*, ¶ [0032], FIG. 6 (step 622).

As highlighted in Figure 6 of Sorenson on the right, the combination of steps 616, 620, and 618—in which Sorenson determines whether the resolved IP address is on the IP backlist and denies the connection if it is—corresponds to "discarding the service request packet if the [resolved] service server IP address does not belong to a preset service server IP address … in a preset list," as recited in elements [1.3], [6.3], and [11.3]. Ex. 1003, ¶¶ 220-221.


IP DEVICE CALL INITIATION
605
USER ACTIVATES IP DEVICE CALL REQUEST
606
608
DOMAIN NAME DIALED ?
N
Y
610
IP DEVICE COMPARES DOMAIN NAME AGAINST BLACKLIST
612
DOMAIN NAME ON BLACKLIST ?
N
Y
614
IP DEVICE RESOLVES DOMAIN NAME TO IP ADDRESS
616
discarding if the resolved IP address "does [] belong to a preset service server IP address … in a preset list"
620
Y
N
618
622
IP DEVICE ALLOWS CALL COMPLETION
END
END

2. Rationale to Combine Sorenson with Treuhaft

A POSA would have found it obvious and been motivated to use a list of authorized/unauthorized IP addresses in Treuhaft based on Sorenson's disclosure for several reasons. Ex. 1003, ¶¶ 222-231. Indeed, the combination merely involves the use of known technique (Sorenson's IP address check) to improve similar devices (Treuhaft's system) in the same way and/or applying a known technique (Sorenson's IP address check) to a known system (Treuhaft) ready for improvement to yield predictable results. *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727 (2007); MPEP § 2143.

Treuhaft and Sorenson are similar in several ways. For example, Treuhaft aims to deny DNS queries seeking access to sites "categorized as an adult web site, a potential phishing or pharming site, and a website whose content has been deemed inappropriate by the user or containing material illegal in the country of the user." Ex. 1008, ¶ [0027]; *see also id.* ¶¶ [0006], [0028]. Similarly, Sorenson seeks "to prevent access by user 14 to unauthorized or blacklisted services" on the Internet. Ex. 1009, ¶ [0023]. Thus, Treuhaft and Sorenson are analogous references having the same purpose to prevent users from accessing inappropriate sites, services, or other online resources. Ex. 1003, ¶ 222.

Treuhaft and Sorenson also have similar structure and operation. *Id.*, ¶ 223. For example, Treuhaft's DNS name server 120 stores subscriber information 280 used to control access to authorized/unauthorized sites for each user or subscriber of the system. *See, e.g.*, Ex. 1008, ¶¶ [0023], [0028], [0029], [0034], [0036], [0039], [0064], [0065]. Similarly, Sorenson's system maintains an IP address 510 and domain name 512 blacklist 500 used to control user access to online resources. *See* Ex. 1009, ¶¶ [0023], [0025], [0026], [0028]-[0032]. Both systems also receive and screen connection requests before or allowing or denying them. *Compare* Ex. 1008, ¶¶ [0064]-[0067], FIG. 5B *with* Ex. 1009, ¶¶ [0031]-[0032], FIG. 6. Accordingly, Treuhaft and Sorenson describe structurally- and functionality- similar systems designed to achieve a similar purpose. Ex. 1003, ¶ 223.

To the extent it is argued that Treuhaft only checks the <u>domain name</u> before allowing or denying the connection, Sorenson improves upon Treuhaft by using a blacklist to check <u>both</u> the domain name <u>and</u> the IP addressed resolved from that domain name before allowing or denying a connection. Ex. 1009, ¶¶ [0027], [0031], [0032]; Ex. 1003, ¶ 224. A POSA would have understood that domain names and IP addresses do not necessarily have a one-to-one correspondence, as multiple domain names might resolve to the same IP address. Ex. 1003, ¶ 225. In some cases, for example, the same inappropriate website may have multiple domain names, or different DNS servers may resolve different domain names to that same inappropriate address. *Id.* An administrator might be aware of some of those domain names but not others, and thus only include the known domain names in a domain name blacklist. *Id.* If a user later issues a connection request using one of the unknown domain names for the inappropriate site not on the blacklist, the system would allow the connection to the inappropriate site. *Id.* Sorenson's technique of checking the resolved IP address as well as the domain name, however, would catch and deny such inappropriate connection requests that Treuhaft's system might otherwise allow. *Id.* Accordingly, Sorenson's application of both a domain name and an IP address blacklist improves upon Treuhaft, to the extent Treuhaft only discloses checking the domain name before allowing or denying a connection. *Id.* Thus, a POSA would have had motivation to make the combination.

Adding Sorenson's technique of applying an IP address blacklist (in addition to a domain name blacklist) to Treuhaft would improve Treuhaft in the same way it improves Sorenson's system. Ex. 1003, ¶ 226. Sorenson receives a connection request including a domain name, compares the domain name against the blacklist, resolves the domain name into an IP address if the domain name is not on the blacklist, compares the resolved IP address against the IP address blacklist, and then allows the connection if the IP address is not on the blacklist. *See* Ex. 1009, ¶ [0031]-[0032], FIG. 6 (steps 608, 610, 612, 614, 616, 621, 622). Similarly, Treuhaft's name server receives a DNS query containing a domain name, resolves the domain name to its corresponding IP address, and applies the subscriber information to determine whether to return the resolved IP address or block that IP address and return a different one. Ex. 1008, ¶ [0064]-[0065]; FIG. 5B (steps 530-540).

Because Treuhaft and Sorenson follow this same general sequence in processing a request, Sorenson's step of checking the IP address could be included as part of step 540 of Treuhaft's method 500 or added after step 540 as another step. *See* Ex. 1008, ¶ [0065], FIG. 5B; Ex. 1003 ¶ 227. In step 540, Treuhaft resolves the domain name in the DNS query to its corresponding IP address and determines whether to use that IP address or another IP address in the DNS response. Ex. 1008, ¶ [0065]. Having the resolved IP address, the DNS name server 120 of Treuhaft could simply check that IP address against Sorenson's added blacklist at that time as part of the decision whether to return it or return another address to the host device 105. Ex. 1003, ¶ 227. Accordingly, Sorenson's known technique of checking the resolved IP address could be incorporated into Treuhaft's similar system to improve Treuhaft in the same way it benefits Sorenson. *Id.*

Treuhaft stands ready for improvement by adding Sorenson's technique, and the combination would have been made through routine skill in the art with predictable results. *Id.* ¶ 228. For example, Treuhaft's method 500 stands ready for improvement by adding Sorenson's IP address check as part of step 540, or another step following step 540, as discussed above. *Id.* Since Treuhaft has already resolved the domain name into its IP address in step 540, the modified system would simply check the resolved IP address against the blacklist at that time. *Id.* Thus, a POSA would have incorporated Sorenson's technique without otherwise significantly modifying or redesigning Treuhaft's method. *Id.*

Moreover, Treuhaft's DNS name server 120 already has a memory 200 storing subscriber information 280. *See* Ex. 1008, ¶¶ [0033]-[0034], FIG. 2. The subscriber information 280 likewise would be augmented to include Sorenson's blacklist information for each user or subscriber without otherwise significantly changing the structure or operation of the DNS name server 120. Ex. 1003, ¶ 229. Thus, the combination would yield predictable results and would be made with a reasonable expectation of success. *Id.* Accordingly, a POSA would have found it obvious to combine Treuhaft and Sorenson as proposed. *Id.*

As explained above with respect to elements [1.3] and [1.4], Treuhaft discloses, or at least suggests, that its subscriber information 280 constitutes a preset list of authorized and/or unauthorized IP addresses. *Supra* Section II.B.1.d, e. Thus, even though Sorenson discloses applying an IP address <u>blacklist</u> to discard the request if the resolved IP address is <u>on</u> the list--rather than "does <u>not belong</u>" (emphasis added) to the list, as claimed, Treuhaft discloses or renders obvious applying a <u>whitelist</u> of authorized IP addresses, a blacklist of unauthorized IP addresses, or both in controlling the host device's access to the Internet. *Id.*; Ex. 1003, ¶ 230. And, in the case of a <u>whitelist</u> of <u>authorized</u> IP addresses, Treuhaft discards the DNS query if the resolved IP address is <u>not</u> on the whitelist in the subscriber information 280. Ex. 1003, ¶ 230. Thus, Treuhaft in combination with Sorenson renders obvious "discarding the service request packet if the [resolved] service server IP address does <u>not belong</u> to a preset service server IP address … in a preset list," as recited in elements [1.3], [6.3], and [11.3]." *Id.* And for the same reasons, the combination of Treuhaft and Sorenson renders obvious that the preset list lists "service server IP addresses <u>under an access authority</u> of the terminal device" as recited in elements [1.4], [6.4], and [11.4], rather than <u>not under an access authority</u> of the terminal device. *Id.*

As discussed above, Treuhaft discloses or at least suggest implementing the subscriber information 280 as a "list" of authorized/unauthorized IP addresses. *Supra* Sections II.B.1. d, e; Ex. 1003, ¶ 231. To the extent it is argued that Treuhaft alone does not expressly disclose or suggest a list, however, the combination with Sorenson's disclosure of an IP address blacklist further renders obvious using a "list" to implement Treuhaft's authorized/unauthorized IP addresses. *Id.*, ¶ 231.

Accordingly, Treuhaft in view of Sorenson renders obvious elements [1.3], [1.4], [6.3], [6.4], [11.3], and [11.4], and so the combination of Treuhaft in view of Sorenson further renders obvious claims 1, 4-6, and 9-11 of the '040 Patent.

D. Grounds 5 and 6: Treuhaft/Sorenson in View of Bellinson Renders Obvious Claims 1, 4-6, and 9-11 of the '040 Patent Under § 103

Potentially, Patent Owner may contend that Treuhaft, or Treuhaft and Sorenson only disclose or suggest applying an IP address blacklist of unauthorized IP addresses, rather than a whitelist of authorized IP addresses, in determining whether to discard the DNS query. Thus, Patent Owner may contend that Treuhaft, or Treuhaft and Sorenson disclose or suggest discarding the service request packet if the resolved IP address is on the list, rather than "does not belong to a preset service server IP address … in a preset list," as recited in elements [1.3], [6.3], and [11.3]. For similar reasons, Patent Owner may also contend that Treuhaft's list contains IP addresses not under the authority of the host device 105, rather than "service server IP addresses under an access authority of the terminal device" (emphasis added) as recited in elements [1.4], [6.4], and [11.4]. Patent Owner would be incorrect at least for the reasons discussed above. *Supra* Section II.B.1.e.

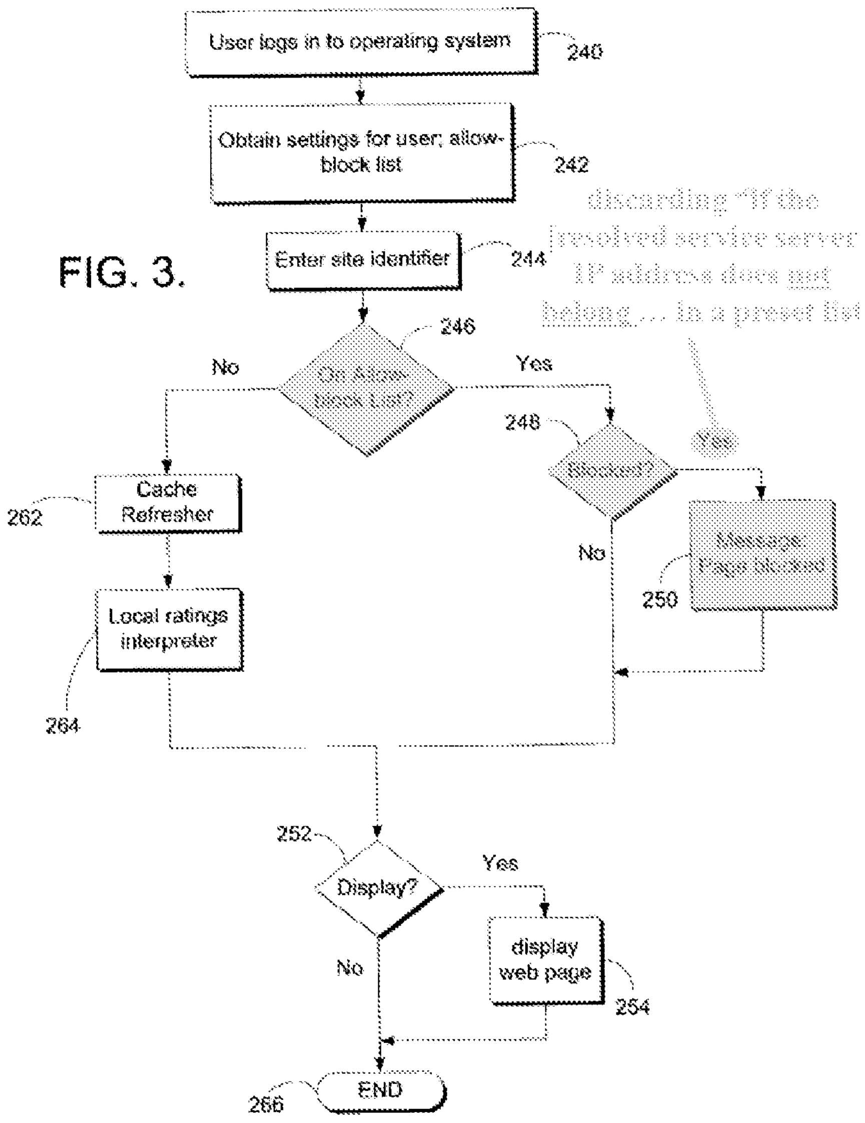
Even if Patent Owner were correct, however, Bellinson further discloses or suggests applying an allow-block list (i.e., a whitelist, blacklist, or combination of both) in determining whether to discard the request or establish a connection, and thus discloses these elements. Ex. 1003, ¶ 233. As explained below, Treuhaft in view of Bellinson, and Treuhaft and Sorenson in view of Bellinson further render obvious claims 1, 4-6, and 9-11 of the '040 Patent.

1. Bellinson Discloses Discarding the Request "if the [Resolved] Service Server IP Address Does Not Belong … in a Preset List" (Elements [1.3], [6.3], and [11.3])

Bellinson discloses elements [1.3], [6.3], and [11.3] by blocking a connection request if a requested site address is not found on an allow-block list (claimed preset list). Ex. 1003, ¶ 234. Specifically, Bellinson discloses "a system and method for controlling whether a user may access certain Internet sites" by applying "an allow-block list" to "determine[] whether the URL is referenced on the allow-block list and, if so, allow[] or disallow[] access to the site referenced by the URL accordingly." Ex. 1010, Abstract. In Bellinson, "[t]he allow-block list is a listing of specific site identifiers that the user is <u>expressly authorized to view</u> or prohibited from viewing." *Id.*, ¶ [0020]; *see also id.*, ¶¶ [0009] ("The allow-block list is a file containing a listing of specific URLs that the user is <u>expressly authorized to view</u> or expressly prohibited from viewing.").

As shown in Figure 3 (reproduced below), in step 244 Bellinson's system receives an access request containing "a specified site identifier that references an Internet site. Examples of such site identifiers include designators such as www.microsoft.com but could also include an Internet Protocol (IP) address." *Id.*, ¶ [0049], FIG. 3. In step 246 of Figure 3, Bellinson "determines whether the site identifier is on the allow-block list at step 246. *Id.* "If the site identifier is referenced on the allow-block list," in step 248 Bellinson "determine[s] whether the site identifier is designated as blocked on the allow-block list." *Id.*, ¶ [0050]. "If the site identifier is [designated as] blocked," Bellinson blocks the connection. *Id.* Otherwise, Bellinson allows the connection. *Id.*, ¶ [0051].

As highlighted in Figure 3 below, Bellinson discloses discarding the request "if the [resolved service server IP address does not belong … in a preset list," as claimed, by determining that the allow-block list does not designate the site identifier as allowed in steps 246 and 248. *Id.*, ¶ [0050]; Ex. 1003, ¶ 236. The allow-block list contains a list of site identifiers designated as allowed and a list of site identifiers designated as blocked. Ex. 1010, ¶¶ [0020] ("[t]he allow-block list is a listing of specific site identifiers that the user is expressly authorized to view or prohibited from viewing."), [0009]; Ex. 1003, ¶ 236. Thus, in determining to block access to a site, Bellinson determines that the site identifier is not found on the allow portion of the list, because the block portion of the list designates the site identifier as blocked. Ex. 1003, ¶ 236. By blocking the request if the address is not on the allow portion of the allow-block list, Bellinson discloses discarding the request "if the [resolved service server IP address does not belong … in a preset list," as recited in elements [1.3], [6.3], and [11.3]. Ex. 1003, ¶ 236.


User logs in to operating system
240
Obtain settings for user; allow-block list
242
discarding "if the [resolved service server IP address does not belong … in a preset list
Enter site identifier
244
246
No
Yes
On Allow-block List?
248
Blocked?
Yes
Cache Refresher
262
Message Page blocked
No
250
Local ratings interpreter
264
252
Display?
Yes
display web page
No
254
END
266
FIG. 3.

2. Rationale to Combine Bellinson with Treuhaft and/or Treuhaft/Sorenson

A POSA would have found it obvious and been motivated to implement Bellinson's allow-block list in Treuhaft for several reasons. Ex. 1003, ¶¶ 237-248. For example, the references provide teaching, suggestion, and/or motivation for making this combination. *KSR*, 127 S. Ct. 1727 (2007); MPEP § 2143. As explained above, Treuhaft's DNS name server 120 applies the subscriber information 280 for each user or subscriber to, among other things, block access to sites "categorized as an adult web site, a potential phishing or pharming site, and a website whose content has been deemed inappropriate by the user or containing material illegal in the country of the user." Ex. 1008, ¶ [0027]; *see also id.* ¶¶ [0006], [0028]. But Treuhaft does not block all connections—only those seeking access to unauthorized sites. Ex. 1003, ¶ 234. Thus, Treuhaft contemplates both authorized and unauthorized sites, and Bellinson's allow-block list mechanism makes it possible to specifically designate both authorized and unauthorized sites. *Id.* Accordingly, a POSA would have sought to incorporate Bellinson's allow-block list in Treuhaft as a mechanism to allow and block access to sites respectively deemed authorized and unauthorized. *Id.*

Additionally, whitelists—like the allow portion of Bellinson's list, and blacklists—like the block portion of Bellinson's list, were known and used interchangeably and/or together in the same system to control access to sites. Ex. 1003, ¶ 238; Ex. 1011, 3[10]. Depending on the particular implementation, the administrator may find a whitelist, a blacklist, or a combination of both appropriate for a given situation. Ex. 1003, ¶ 238; Ex. 1011, 3. For example, in some cases, the administrator may desire a strict approach to access control in which only connections to certain authorized sites are permitted, and thus may choose a whitelist. Ex. 1003, ¶ 238. This approach may be useful, for example, in systems with child, student, and/or employee users that should only access certain specific websites or other resources. *Id.* In other cases, the administrator may want to give users more leeway to access a variety of sites while blocking access to certain known unauthorized sites, and thus choose to use a blacklist. *Id.* This approach may be useful, for example, if the administrator is more concerned with preventing malicious attacks on the user's computer rather than with preventing the user from accessing certain types of content. *Id.*

[10] For ease of reference, Requester cites the PDF page number of Exhibit 1002.

The extent of an administrator's knowledge may also factor into the decision of whether to use a whitelist and/or a blacklist. Ex. 1003, ¶ 239; Ex. 1011, 3. For example, if the administrator has complete knowledge of the sites a user must or should access in the particular environment, a whitelist may be appropriate. Ex. 1003, ¶ 239. But in cases where the administrator does not have such knowledge, a blacklist may be more appropriate. *Id.* Accordingly, incorporating Bellinson's allow-block list into Treuhaft gives administrators more options for controlling users' or subscribers' access to the Internet, and thus would have sought to make the combination. *Id.* Thus, a POSA would have had motivation to make the combination.

The combination of Bellinson with Treuhaft merely involves using a known technique (Bellinson's allow-block list) to improve a similar device (Treuhaft's system) in the same way and/or applying a known technique (Bellinson's allow-block list) to a known system (Treuhaft) ready for improvement to yield predictable results. *KSR Int'l Co. v. Teleflex Inc.*, 127 S. Ct. 1727 (2007); MPEP § 2143. Treuhaft and Bellinson describe systems similar in a number of ways. As discussed, Treuhaft allows an administrator to use subscriber information for users or subscribers of the system to block access to sites "categorized as an adult web site, a potential phishing or pharming site, and a website whose content has been deemed inappropriate by the user or containing material illegal in the country of the user" Ex. 1008, ¶ [0027]; *see also id.* ¶¶ [0006], [0028]. Similarly, Bellinson's technique using the allow-block list enables "parents to effectively control a child's web site access." Ex. 1010, ¶ [0008]; *see also id.*, ¶¶ [0003]-[0007], [0009], [0039]-[0041], [0048], [0057], [0058]. Thus, Treuhaft and Bellinson are analogous references with the same purpose of allowing administrators to prevent users from accessing inappropriate sites, services, or other online resources. Ex. 1003, ¶ 240.

Treuhaft and Bellinson also have similar structure and operation. As discussed previously, Treuhaft's administrator inputs subscriber information 280 for a particular user or subscriber into the DNS name server 120, which the DNS name server 120 later applies to control that user's or subscriber's access to the Internet. *See, e.g.*, Ex. 1008, ¶¶ [0023], [0028], [0029], [0034], [0036], [0039], [0064], [0065]. Likewise, in Bellinson, "an administrator or parent would supply the content settings service with settings for a specified user," and those "settings could include the user's age group, age group map and an allow-block list." Ex. 1010, ¶ [0057]. Later, when that user sends a connection request including a site identifier, the Bellinson system retrieves the user's settings, including the allow-block list, and applies the allow-block list in determining whether to allow or block the connection to that site. *See, e.g., id.*, ¶¶ [0039], [0049]-[0051]. Accordingly, Treuhaft and Bellinson describe structurally- and functionality- similar systems designed to achieve a similar purpose. Ex. 1003, ¶ 241.

Adding Bellinson's allow-block list to Treuhaft would improve Treuhaft in the same way it improves Bellinson's system. Ex. 1003, ¶ 242. When receiving a connection request including a site identifier, Bellinson retrieves the user's previously-stored settings including the allow block list, Ex. 1010, ¶ [0039], FIG. 3 (step 242), and compares the site identifier to the allow-block list in determining whether to allow or block the connection to that site, *id.*, ¶¶ [0039], [0049]-[0051]. Similarly, when Treuhaft's DNS name server 120 receives a DNS query containing a domain name, it retrieves the previously-stored subscriber information for the user or subscriber and applies the subscriber information in determining how to respond to the DNS query. Ex. 1008, ¶ [0064]-[0065]; FIG. 5B (steps 530-540).

Because Treuhaft and Bellinson follow this same general sequence in processing a request, Bellinson's allow-blocklist would be incorporated into Treuhaft's process to improve Treuhaft in substantially the same way. Ex. 1003, ¶ 243. For example, in the combined system, Bellinson's step of retrieving the user's settings including the allow-block list could be performed as part of Treuhaft's step 535 when the DNS name server 120 retrieves the user's subscriber information. *Compare* Ex. 1010, ¶ [0039], FIG. 3 (step 242) *with* Ex. 1008, ¶ [0064], FIG. 5B (step 535); *see also* Ex. 1003, ¶ 243. Additionally, Bellinson's steps of applying the allow-block list would be performed as part of Treuhaft's corresponding step of applying the subscriber information to process the DNS query. *Compare* Ex. 1010, ¶¶ [0049]-[0051] , FIG. 3 (steps 246, 248) *with* Ex. 1008, ¶ [0065], [0066), FIG. 5B (steps 540, 545); *see also* Ex. 1003, ¶ 243. Because the DNS name server 120 of Treuhaft has already resolved the IP address at this point, it would simply check that IP address against Bellinson's allow-block list as part of the decision whether to return it or return another address to the host device 105. Ex. 1003, ¶ 243. Accordingly, Bellinson's known technique of applying an allow-block list would be incorporated into Treuhaft's similar system to improve Treuhaft in the same way it benefits Sorenson. *Id.*

Treuhaft stands ready for improvement by adding Bellinson's technique, and the combination would be made through routine skill in the art with predictable results. *Id.* ¶ 244. For example, Treuhaft's method 500 stands ready for improvement by adding Bellinson's allow-block list process, as discussed above. *Id.* And Since Treuhaft has already resolved the domain name into its IP address in step 540 or 545, the modified system would simply check the resolved IP address against the allow-block at that time. *Id.* Thus, a POSA would have incorporated Bellinson's technique without otherwise significantly modifying or redesigning Treuhaft's method. *Id.*

Moreover, Treuhaft's DNS name server 120 already has a memory 200 storing subscriber information 280. *See* Ex. 1008, ¶¶ [0033]-[0034], FIG. 2. The subscriber information 280 likewise could be augmented to include Bellinson's allow-block list information for each user or subscriber without otherwise significantly changing the structure or operation of the DNS name server 120. Ex. 1003, ¶ 245. Thus, the combination would yield predictable results and would be made with a reasonable expectation of success. *Id.* Accordingly, a POSA would have found it obvious to combine Treuhaft and Sorenson as proposed. *Id.*

It is noted that combination of Treuhaft and Bellinson additionally renders obvious the recitation in elements [1.4], [6.4], and [11.4] that the preset list contains "service server IP addresses <u>under an access authority</u> of the terminal device." Specifically, the <u>allow</u> portion of Bellinson's allow-block list, incorporated into Treuhaft, is a list of site identifiers under an access authority of Treuhaft's host device 105 (claimed terminal device). Ex. 1003, ¶ 246.

As discussed above, Treuhaft discloses or at least suggest implementing the subscriber information 280 as a "list" of authorized/unauthorized IP addresses. *Supra* Section II.B.1. d, e; Ex. 1003, ¶ 247. To the extent it is argued that Treuhaft alone does not expressly disclose or suggest a list, however, the combination with Bellinson's disclosure of an allow-block <u>list</u> further renders obvious using a "list" to implement Treuhaft authorized/unauthorized IP addresses. *Id.*, ¶ 247.

Accordingly, for the reasons above, the combination of Treuhaft in view of Bellinson and/or the combination of Treuhaft/Sorenson in view of Bellinson, renders obvious elements [1.3], [1.4], [6.3], [6.4], [11.3], and [11.4]. Thus, these combinations render obvious claims 1, 4-6, and 9-11 of the '040 Patent.

E. Secondary Considerations

This Request demonstrates that the Challenged Claims of the '040 Patent are unpatentable as anticipated and obvious in view of the prior art references. The Applicant did not identify any evidence of secondary considerations during prosecution. Further, the clear teachings in the prior art cannot be overcome by any supposed "secondary considerations." *Graham v. John Deere Co.*

of Kansas City, 383 U.S. 1, 36 (1966); *see also* Ex. 1003, ¶ 248.

III. DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS

Based on on information available to Requester, the '040 Patent is the subject of ten District Court litigations, as listed in the below table. Requester is unaware of any prior reexaminations or other post-grant proceedings involving the '040 Patent.

Title	Case No.	Court	Filed	Status
Orbit Licensing LLC v. The Mathworks, Inc.	1-21-cv-01089	D.Del.	2021-7-28	Pending
Orbit Licensing LLC v. Velocix Solutions USA Inc.	1-21-cv-01090	D.Del.	2021-7-28	Pending
Orbit Licensing LLC v. Wowza Media System, LLC	1-21-cv-00953	D.Del.	2021-7-30	Pending
Orbit Licensing LLC v. Tencent America LLC	1-21-cv-00962	D.Del.	2021-7-30	Terminated 2021-10-1
Orbit Licensing LLC v. OnApp, Inc.	1-21-cv-00964	D.Del.	2021-7-30	Pending
Orbit Licensing LLC v. EidosMedia, Inc.	1-21-cv-05691	S.D.N.Y.	2021-7-30	Pending
Orbit Licensing LLC v. Akamai Technologies, Inc.	1-21-cv-00938	D.Del.	2021-7-29	Terminated 2021-9-20
Orbit Licensing LLC v. CodePen, Inc.	1-21-cv-00943	D.Del.	2021-7-29	Pending
Orbit Licensing LLC v. Limelight Networks, Inc.	1-21-cv-00948	D.Del.	2021-7-29	Pending
Orbit Licensing LLC v Red Hat, Inc.	1-21-cv-00949	D.Del.	2021-7-29	Pending

IV. CONCLUSION

Requester requests the Office to issue an office action rejecting the challenged claims based on Grounds 1-6 described above. The Commissioner is hereby authorized to charge Deposit Account 12-0769 under Docket No. U022-0007RE the *Ex Parte* Reexamination fee of $12,000 under 37 C.F.R. § 1.20(c)(1). Requester believes no other fee is due with this submission; however, the Commissioner is hereby authorized to charge any fee deficiency or credit any over-payment to Deposit Account 12-0769.

Please direct all correspondence in this matter to the undersigned.

Dated: November 9, 2021 Respectfully submitted,

By: */James D. Stein/*

James D. Stein (Reg. No. 63,782)
Lee & Hayes, P.C.
75 14th Street NE, Suite 2500
Atlanta, Georgia 30309
james.stein@leehaye.com
P: 040.736.1918

Counsel for Requester Unified Patents, LLC

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Jiancheng Guo et al.	§	Attorney Docket No.: U022-0007RE
	§	
U.S. Patent 9,578,040	§	
	§	
Issue Date: February 21, 2017	§	Customer No.: 29,150
	§	
Filing Date: December 16, 2014	§	
	§	
For: PACKET RECEIVING METHOD, DEEP PACKET INSPECTION DEVICE AND SYSTEM	§ § §	

Mail Stop "*Ex Parte* Reexam"
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§ 1.248 and 1.550 and M.P.E.P. § 2266.03, the undersigned attorney for the Requester certifies that a copy of the foregoing **REQUEST FOR *EX PARTE* REEXAMINATION OF U.S. PATENT 9,578,040** and accompanying Exhibits Ex. 1001 to 1016 was served by Express Mail at the following address of record for the subject patent:

Gerald T. Gray
Leydig, Voit & Mayer, Ltd.
(for Huawei Technologies Co., Ltd)
Two Prudential Plaza Suite 4900
180 North Stetson Avenue
Chicago IL 60601

Dated: November 9, 2021

By: */James D. Stein/*
James D. Stein (Reg. No. 63,782)