proxy (i.e., the "secure server"). The DNS proxy module or DNS module of the modified C-HTTP name server determines whether the requested host is a member of the closed network. If the requested host is a member of the closed network (i.e., a server-side proxy), then a C-HTTP name service response is returned to the client-side proxy including an IP address for the server-side proxy and Nonce values, and a secure, encrypted channel is automatically initiated/created between the client-side proxy and the server-side proxy. However, if the requested host is not a member of the closed network, then the DNS proxy module or DNS module performs a lookup to the conventional DNS server and returns the IP address to the client-side proxy.

As set forth in Ex.1009 ¶¶ 44-47 and Appendix C, it would have been apparent to a person of ordinary skill in the art as a mere design choice to consolidate domain name resolution functions in Kiuchi's C-HTTP name server. Kiuchi clearly recognizes and discloses that a conventional DNS lookup for the domain name is needed when access is not being requested to a secure server, i.e., when the requested server-side proxy is not registered in the closed network. (Ex.1002 at 65, sec. 2.3(2)). This is identical to the '151 Patent, where a DNS lookup is performed when access is not being requested to a secure server. (Ex.1001 at 38:12-16).

Kiuchi defines three new components for the system, namely the client-side proxy, the server-side proxy, and the C-HTTP name server. (Ex.1002 at 64, sec. 2.1). While Kiuchi describes a system in which a conventional DNS lookup request is made from the client-side proxy, it would have been apparent to a person of ordinary skill in the art based on Kiuchi's teachings to make the conventional DNS lookup request from the C-HTTP name server. As discussed above, the C-HTTP name server already determines whether the DNS request received from the client-side proxy is requesting access to a secure server (i.e., a server-side proxy). Rather than returning an error status to the client-side proxy when the DNS request is not requesting access to a secure server, it would have been trivial and obvious as a mere design choice for the C-HTTP name server to pass the domain name received in the C-HTTP name service request to the conventional DNS server (i.e., a "DNS function"), as depicted in the following Figure:

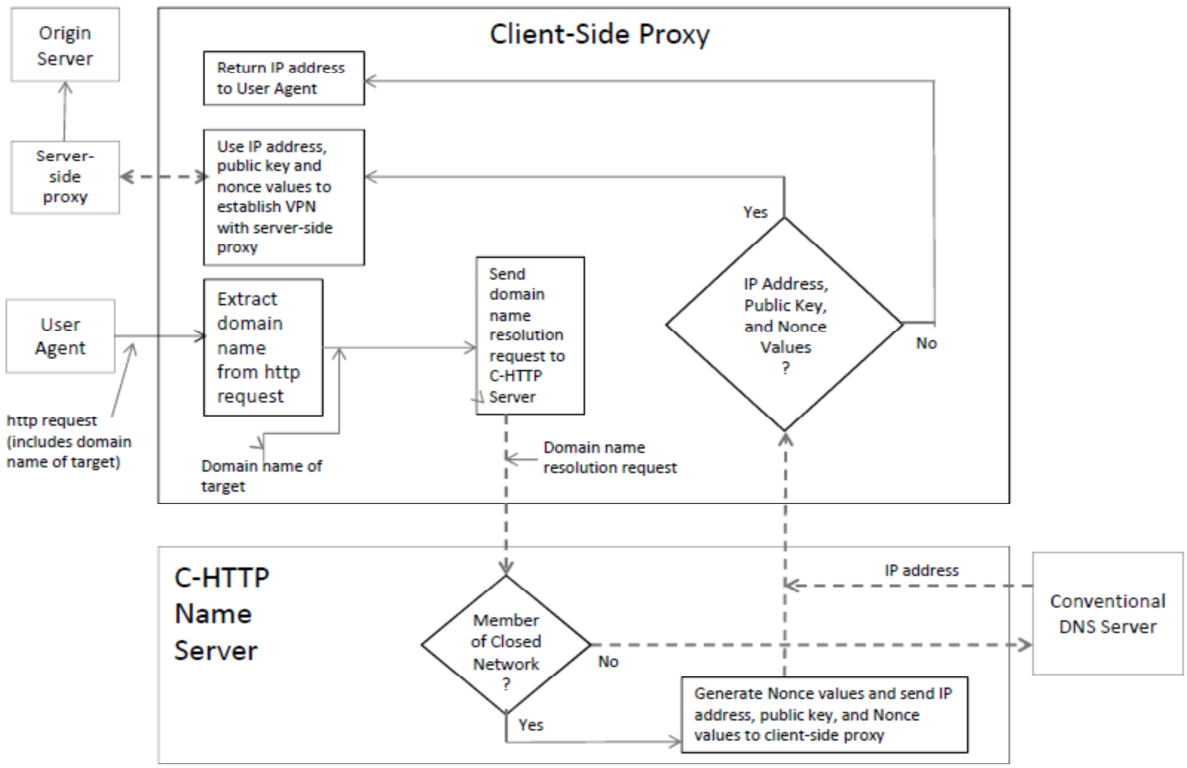

Figure 3 – Alternate Functional Diagram of Kiuchi

Such a configuration, which places a DNS proxy server function in a modified C-HTTP name server (similar to placement of the DNS proxy server function of the '151 patent in the DNS server – see Ex.1001 at FIG. 26), is merely a rearrangement of existing functions within the C-HTTP system and could be implemented with little or no modification to Kiuchi's protocols. For example, a C-HTTP name service response message containing an IP address without a public key and Nonce values (e.g., using values of zero or other convention for the public key and Nonce fields, or modifying the protocol to use a previously unused flag in the response to indicate that a public key and Nonce values are not provided) would indicate to the client-side proxy that the DNS request is not requesting

access to a secure server and hence that no secure/encrypted channel is needed. The motivation for modifying Kiuchi in this way would have been to streamline the operation of the system, e.g., instead of having the C-HTTP name server send an error status to the client-proxy which would in turn initiate a conventional DNS inquiry, the modification eliminates the error status message from the process by having the C-HTTP name server directly initiate the request to the conventional DNS server. Thus, the consolidation of domain name resolution functions in the C-HTTP name server is obvious in view of Kiuchi. See *KSR Intern. Co. v. Teleflex Inc.*, 550 U.S. 398, 401; 127 S.Ct. 1727, 173 (2007) ("a combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results"); *Sakraida v. Ag Pro, Inc*., 425 U.S. 273, 282, 96 S.Ct. 1532 (1976) (when a patent "simply arranges old elements with each performing the same function it had been known to perform" and yields no more than one would expect from such an arrangement, the combination is obvious).

The modified C-HTTP name server includes a "DNS proxy module" because it contains a program that responds to a domain name inquiry in place of a DNS (i.e., it responds to a DNS request in place of a conventional DNS server when the requested host is a member of the closed network). The modified C-HTTP name server also includes a "DNS module" because it includes a program that performs a lookup service and returns an IP address for a requested domain

name (i.e., it first performs a lookup to determine if the requested host is registered in the closed network, and then, if needed, performs a lookup to the conventional DNS server); in Kiuchi, this DNS module also performs the functions of a DNS proxy by responding to a DNS request in place of the conventional DNS server when the requested host is a member of the closed network, as discussed above.

The C-HTTP is a data processing device having a memory storing a "DNS proxy module" or "DNS module" that intercepts DNS requests sent by the client-side proxy (i.e., the "client") and performs the steps recited in the claims. The client-side proxy is a "client" because it is a computer that sends a DNS request. The server-side proxy is a "secure server" because it is a server that requires authorization for access and can communicate in an encrypted channel.

Ground 1. Claim 1 is Obvious Over Kiuchi

As set forth in Ex.1009 ¶¶ 29-47 and Appendix C, Kiuchi teaches all of the limitations of claim 1. With regard to claim 1, the modified C-HTTP name server including a DNS proxy module is a data processing device. The functions performed in the modified C-HTTP name server are necessarily stored in computer program code in memory, as is the case for any such processing device. (Ex.1009 ¶ 44).

When the client-side proxy (i.e., the "client") receives the HTTP request from the user agent, it generates (and transmits) a DNS request that is sent to the

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.