IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

Bedrock Computer Technologies LLC,	
Plaintiff,	
v.	Case No. 6:09-CV-269-LED
Softlayer Technologies, Inc., et al., Defendants.	JURY TRIAL DEMANDED

DEFENDANTS' MOTION FOR SUMMARY JUDGMENT OF INVALIDITY OF U.S. PATENT NO. 5,893,120

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Defendants Amazon.com Inc. ("Amazon"), AOL Inc. ("AOL"), Google, Inc. ("Google"), Match.com ("Match.com"), MySpace, Inc. ("MySpace"), SoftLayer Technologies, Inc. ("SoftLayer"), and Yahoo! Inc. ("Yahoo!"), collectively referred to as "Defendants," move this Court for summary judgment of invalidity of U.S. Patent No. 5,893,120 ("the '120 Patent"). All claims of the '120 Patent are anticipated by Linux source code that was published more than one year before the filing date of the '120 Patent.

I. INTRODUCTION AND SUMMARY OF ARGUMENTS

This motion is made simple by Bedrock's infringement contentions where Bedrock has accused the Linux Operating System of infringing the '120 Patent. The Linux Operating System was first released as early as 1991, well before the filing date of the '120 Patent. Bedrock does not accuse the entire Linux Operating System of infringing the '120 Patent -- just a file called "route.c." Furthermore, Bedrock does not accuse servers running all versions of the Linux Operating System -- just the servers running later versions. However, the file route.c in the accused versions of the Linux Operating System is not materially different from the prior art versions. Therefore, the Linux code Bedrock alleges practices the '120 patent is virtually identical as the prior art Linux code. Obviously Bedrock cannot have it both ways. It is well established that claims are not "noses of wax"; they must be construed the same for infringement and invalidity purposes. Indeed, Bedrock's expert, Dr. Jones, does not dispute that the majority of the claim limitations are met by the Prior Art Linux Code. In light of Bedrock's positions with respect to infringement, there is no genuine issue of material fact that prior art versions of route.c anticipate all of the claims of the '120 Patent.

II. STATEMENT OF UNDISPUTED FACTS

A. THE '120 PATENT

1. The priority date of the '120 Patent is its filing date, January 2, 1997.

2. Identifying and deleting records from a database were well-known in the art at the time of the invention of the ’120 Patent (*see* Exh. G, Nemes Dep. 743:7-13 (Dec. 30, 2010)).

3. The concept and practice of storing and deleting records in an external chain were well-known in the art at the time of invention of the ’120 Patent (*see id*. at 743:14-20).

4. The concept and practice of removing a record from an external chain for deletion was well-known in the art at the time of the invention of the ’120 Patent (*see id*. at 743:21-24).

5. It was well-known in the prior art to store information or data in a computer-controlled storage mechanism by utilizing a key value stored in the records (*see* Exh. F, ’120 Patent at col. 1, ll. 23-27).

6. The “hashing” technique is another well-known method of storing and retrieving information from computer storage (*see id*. at 1:34-46). A disadvantage to hashing is “collisions,” where more than one key will translate to the same storage address (*see id*.)

7. It was well-known in the art to remedy these “collisions” by “linear probing” or “external chaining.” (*see id*. at 1:54-2:6; *see also* Exh. G, Nemes Dep. 741:24-742:6 (Dec. 30, 2010)). It was also well-known to remedy these “collisions” through quadratic and random probing (*see* Exh. G, Nemes Dep. 742:13-25, (Dec. 30, 2010)).

8. Linux kernel version 1.3.51 was publicly available on the Internet on or around December 27, 1995 (*see* DEF00009284-DEF00009285; DEF00007865-DEF00007899; Kuznetsov Dep. at 68:10 – 69:22 (Jan. 27, 2011)). Linux kernel version 1.3.52 was publicly available on the Internet on or around December 29, 1995 (*see* DEF00009284-DEF00009285; DEF00001013-DEF00001043; Kuznetsov Dep. at 41:24 – 44:10 (Jan. 27, 2011)). Linux kernel version 2.0.1 was publicly available on the Internet on or around July 3, 1996 (*see*

DEF00009284-DEF00009285; DEF00008567-DEF00008601 Kuznetsov Dep. at 69:25 – 71:15 (Jan. 27, 2011)).

9. The route.c file in the Prior Art Linux Code contains a function *rt_cache_add()* that adds a record to the route cache (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1299 – 1385).

10. The *rt_cache_add()* function in Linux kernel versions 1.3.51, 1.3.52, and 2.0.1 are substantively identical (*see* Exh. E, Jeffay Decl. Ex. A at ¶ 61 fn. 12).

B. CLAIM CONSTRUCTION

The court construed or adopted certain claim constructions of the elements of the claims in the ’120 Patent in its Memorandum Opinion and Order issued January 10, 2011 (Dkt. No. 369).

III. LEGAL STANDARD

Summary judgment is warranted if the pleadings and discovery show that there is no genuine issue as to any material fact and that the moving party is entitled to judgment as a matter of law. *See* Fed. R. Civ. P. 56(c); *see also Celotex Corp. v. Catrett*, 477 U.S. 317, 327 (1986). Defendants bear the initial responsibility of identifying the absence of a genuine issue of material fact. *See Celotex*, 477 U.S. at 323. The burden then shifts to Plaintiff to set forth specific facts demonstrating a genuine factual issue for trial. *See* Fed. R. Civ. P. 56(e); *see also Matsushita Elec. Indus. Co. v. Zenith Radio Corp.*, 475 U.S. 574, 587 (1986); *Anderson v. Liberty Lobby*, Inc., 477 U.S. 242, 252 (1986).

Plaintiff cannot rest on the mere allegations or denials of the pleadings, but must instead produce specific facts, by affidavit or other evidentiary materials, showing that there is a genuine triable issue. *See Anderson*, 475 U.S. at 248. This Court need only resolve factual issues of controversy in favor of Plaintiff where the facts specifically averred by Plaintiff contradict material facts specifically averred by the Defendants. *See Lujan v. Nat'l Wildlife Fed'n*, 497 U.S. 871, 888 (1990); *see also Alexander v. Eeds*, 392 F.3d 138, 142 (5th Cir. 2004).

"A patent is invalid for anticipation if a single prior art reference discloses each and every limitation of the claimed invention." *Schering Corp. v. Geneva Pharms., Inc.*, 339 F.3d 1373, 1377 (Fed. Cir. 2003). A claim is also invalid if the patented invention would have been obvious to a person having ordinary skill in the art to which the subject matter of the patent pertains, at the time the invention was disclosed. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398 (2007) (citing 35 U.S.C. § 103). "In determining whether the subject matter of the patent is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim." *Id.* at 419.

IV. STATEMENT OF ISSUES TO BE DECIDED

1. Whether the prior art Linux Kernels contain each element of claims 1-8 of the '120 Patent.

V. THE PRIOR ART ANTICIPATES AND/OR RENDERS OBVIOUS ALL CLAIMS OF THE '120 PATENT

A. THE PRIOR ART LINUX CODE

Bedrock has accused servers running Linux versions 2.4.29, 2.4.31, 2.6.9, 2.6.11, 2.6.15, 2.6.16, 2.6.18, 2.6.21, 2.6.22, 2.6.23, 2.6.24, 2.6.26, 2.6.27, 2.6.29, 2.6.32, 2.6.33, and 2.6.34 ("the Accused Linux Code") of infringing certain claims of the '120 Patent.[1] Bedrock has specifically identified code within the source code file route.c as infringing.

[1] Bedrock has accused versions prior to version 2.6.25 of infringing claims 1-4 and 7-8, and has accused versions after 2.6.25 of infringing claims 1-8 of the 120 Patent (*see* Opening Expert Report of Dr. Mark Jones).

Prior to the release of the Accused Linux Code, Linux kernel versions 1.3.51, 1.3.52, and 2.0.1 ("the Prior Art Linux Code") were published and in public use on the Internet on or before the earliest date that the inventor asserts he invented what is claimed in the '120 Patent.[2] Two of these versions—versions 1.3.51 and 1.3.52—were published and in public use more than one year prior to the January 2, 1997 filing date of the '120 Patent. These two versions are § 102(b) prior art. Bedrock has provided no evidence supporting a conception date earlier than the filing date of the '120 Patent.

The Prior Art Linux Code contains a version of the route.c computer source code that is substantially similar to the implementation of route.c appearing in some of the Accused Linux Code.[3] For example,[4] in Linux kernel version 2.0.1, the route cache is implemented as a hash table called ip_rt_hash_table ("the IP route hash table"). In Linux kernel version 2.6.18, the route cache is implemented as a hash table called ip_rt_hash_table. Each element of the hash table in Linux kernel version 2.0.1 (each bucket of the hash table) contains a pointer to a record called rtable (a "routable cache record"). Each element of the hash table in Linux kernel version

[2] Linux kernel version 1.3.51 was published and in public use on the Internet on or around December 27, 1995 (*see* Exh. C, Kuznetsov Dep. at 68:10 – 69:22 (Jan. 27, 2011); *see also* DEF00009284-DEF00009285; DEF00007865-DEF00007899). Linux kernel version 1.3.52 was published and in public use on the Internet on or around December 29, 1995 (*see* Exh. C, Kuznetsov Dep. at 41:24 – 44:10 (Jan. 27, 2011); *see also* DEF00009284-DEF00009285; DEF00001013-DEF00001043). Linux kernel version 2.0.1 was published and in public use on the Internet on or around July 3, 1996 (*see* Exh. C, Kuznetsov Dep. at 69:25 – 71:15 (Jan. 27, 2011); *see also* DEF00009284-DEF00009285; DEF00008567-DEF00008601). For example, Linux kernel version 2.0.1 was published and in public use on the Internet at *ftp://ftp.kernel.org/pub/linux/kernel/v2.0/linux-2.0.1.tar.gz*. Route.h in Linux kernel version 2.0.1 carries a file modification date of July 3, 1996 that indicates that the files downloaded from the kernel.org ftp site have not been modified since July 3, 1996. Further, Alexey Kuznetsov, one of the authors of the route.c code in Linux kernel version 2.0.1, has confirmed that this version was published and in public use on or about July 3, 1996 (*see* Exh. D, Kuznetsov Decl., DEF00009284-85).

[3] The author of the prior art implementation testified that the currently-accused Linux software "does almost exactly the same thing" as the prior art implementation (*see* Exh. C, Kuznetsov Dep. at 96:14-19 (Jan. 27, 2011); *see also id.* at 97:17-19 ("Then, my idea from 1995, which existed in Linux kernel at least for two years, was returned in another incarnation and applied to Linux kernel by Eric Dumazet.")).

[4] For illustrative purposes, the following describes the operation of a portion of the route cache in Linux kernel version 2.0.1 by comparison with the equivalent portions of the route cache implemented in Linux kernel version 2.6.18. However, the same comparison can be made between any of the Prior Art Linux Code and the Accused Linux Code (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 148).

2.6.18 similarly points to a record called an rtable. In both Linux kernel versions 2.0.1 and 2.6.18, the route cache records are linked to one another to form a linked list (an external chain for a hash table bucket). In Linux kernel version 2.0.1, the route hash table is accessed via a hash function called ip_rt_hash_code. In Linux kernel version 2.6.18, the route hash table is accessed via a hash function called rt_hash_code. In both Linux kernel versions 2.0.1 and 2.6.18, this hash function is used to locate a bucket in the route hash table to find (or insert or delete) a specific route cache record (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 149).

Because the Accused Linux Code operates the same way in material respect as the Prior Art Linux Code and because Bedrock asserts that servers running the identified code infringe the ’120 Patent, Bedrock effectively admits that the ’120 Patent is invalid. It is black letter law that claims must be construed the same for infringement and invalidity purposes. *See Amgen Inc. v. Hoeschtst Marion Roussel, Inc.*, 314 F.3d 1313, 1330 (Fed. Cir. 2003) (“[i]t is axiomatic that claims are construed the same way for both invalidity and infringement”). If the accused servers running the Accused Linux Code infringe, then it necessarily follows that the claims are invalid as a matter of law based on the Prior Art Linux Code.

B. THE PRIOR ART LINUX CODE ANTICIPATES ALL CLAIMS OF THE ’120 PATENT

The Prior Art Linux Code independently anticipates each and every claim of the ’120 Patent. The Prior Art Linux Code discloses an information storage and retrieval system wherein route cache records are stored in, and retrieved from, a route cache that is organized as a hash table with collision resolution based on external linked lists (*see, e.g.,* Exh. A, Linux 2.0.1 at route.c, ll. 151; *see also* route.h, ll. 65-81).'[5]

In the Prior Art Linux Code, to add a record to the route cache, the route hash table is used to locate an external chain (a linked list of route cache records) that is then accessed to perform the appropriate task (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 150). Each route cache record maintains a number of data values that describes the state of the record (*id*.). These data values include a reference count that indicates the number of references to the route cache record, and a form of timestamp that indicates when the route cache record was last accessed. When a route cache record is not currently being used by a network connection (when its reference count is zero), and when the record has not been used for a specified period, the route cache record is considered to have expired, according to Bedrock's interpretation of the court's construction of the term expiration (*see, e.g.,* Exh. H, Bedrock's Amended Infringement Contention for Linux kernel version 2.6.18 (Jan. 12, 2011) (hereinafter "Bedrock's Infringement Contention"); *see also* Bedrock's Opening Expert Report of Dr. Mark Jones (hereinafter "Bedrock's Opening Expert Report")), and is eligible for deletion (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 150). When a record is inserted into the route cache, the linked list that comprises an external chain of the IP route hash table is accessed to insert a record into the cache and, while searching for a duplicate of the record just inserted, identify and remove expired route cache records. Therefore, the Prior Art Linux Code discloses a method of "on-the-fly" removal of automatically expiring records from the route cache, which, as described below, is precisely the same as that disclosed in the '120 Patent (*see* Exh. E, Jeffay Decl. Exh. A at ¶¶ 245-247).

---

[5] For convenience, the cited code herein corresponds to Linux 2.0.1. The cited code is identical to the corresponding code in versions 1.3.51 and 1.3.52 (*see* Invalidity Expert Report of Joel Williams at 126). Thus, the analysis herein applies equally to each of versions 1.3.51, 1.3.52, and 2.0.1.

Bedrock’s expert, Dr. Mark Jones, in his rebuttal expert report regarding validity of the ’120 Patent, does not contest the Defendants’ experts’ opinions that a number of the limitations of the ’120 Patent are present in the Prior Art Linux Code (*see* Exh. B, Bedrock’s Rebuttal Expert Report of Dr. Mark Jones (Feb. 1, 2001) (hereinafter “Bedrock’s Rebuttal Expert Report”), Attachment 2 at 2-9). Indeed, Dr. Jones provides no opinions as to whether any of the method claims (claims 3, 4, 7, or 8) are valid in light of the Prior Art Linux Code (*id*.). Accordingly, the Court should grant summary judgment with respect to these claims.

Similarly, Dr. Jones only opines that two of the limitations are not met by the Prior Art Linux Code – “the record search means including a means for identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed” of claims 1 and 5, and the “means for dynamically determining maximum number” of claims 2 and 6. For the reasons discussed below, his opinions do not create an issue of fact.

Below, Defendants demonstrate that the Prior Art Linux Code teaches each and every limitation from the claims of the ’120 Patent.

1. The Prior Art Linux Code Anticipates Claims 1 and 5 of the ’120 Patent

a) The Prior Art Linux Code discloses “[a]n information storage and retrieval system, the system comprising:”

To the extent the preamble is a limitation, the Prior Art Linux Code discloses “[a]n information storage and retrieval system,” as set forth in claims 1 and 5 of the ’120 Patent. The Prior Art Linux Code discloses an information storage and retrieval system for routing related records (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1299 - 1385; *see also* Linux 2.0.1 at route.h). Bedrock’s expert does not contest that the Prior Art Linux Code meets this limitation (*see* Exh. B, Bedrock’s Rebuttal Expert Report, Attachment 2 at 2-9).

b) The Prior Art Linux Code discloses “a linked list to store and provide access to records stored in a memory of the system, at least

some of the records automatically expiring" in claim 1 and "a hashing means to provide access to records stored in a memory of the system and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring" in claim 5.

The Prior Art Linux Code discloses a linked list/hashing table using external chaining to store and provide access to records in a memory of the system as set forth in claims 1 and 5 of the '120 Patent. The Prior Art Linux Code discloses a hash table *ip_rt_hash_table* (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 151; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 248). This hash table consists of *rtable* structures, which are defined as nodes of a linked list and include a pointer to the next record in the linked list of route cache records (*see id*.).

```
65 struct rtable
66 {
67          struct rtable *rt_next;
68          __u32 rt_dst;
69          __u32 rt_src;
70          __u32 rt_gateway;
71          atomic_t rt_refcnt;
72          atomic_t rt_use;
73          unsigned long rt_window;
74          atomic_t rt_lastuse;
75          struct hh_cache *rt_hh;
76          struct device *rt_dev;
77          unsigned short rt_flags;
78          unsigned short rt_mtu;
79          unsigned short rt_irtt;
80          unsigned char rt_tos;
81 };
```

(Linux 2.0.1 at route.h, ll. 65-81.)

Further, under Bedrock's interpretation of the court's construction of expiration, as set forth in Bedrock's infringement contentions and its opening expert report, the Prior Art Linux code discloses that at least some of the records stored in the hash table of route cache records automatically expire as set forth in claims 1 and 5 of the '120 Patent (*see, e.g.*, Exh. H, Bedrock's Infringement Contention at 1-4; *see also* Bedrock's Opening Expert Report at 40-41). Consistent with Bedrock's interpretation of the court's construction of expiration, each route cache record has an associated “lifetime,” after which the record is considered to have expired (*see* Linux 2.0.1 at route.h, ll. 65-81). An example of the test for this expiration is shown below.

```
1369        if ((!rth->rt_refcnt && rth->rt_lastuse + RT_CACHE_TIMEOUT < now)
1370            || rth->rt_dst == daddr
```

(Exh. A, Linux 2.0.1 at route.c, ll. 1369-70).

As shown at line 1369, a record expires when the last use of the route cache record exceeds the timeout value for the route cache record and the route cache record is not currently being referenced. This indicates that the routing cache record is obsolete and therefore no longer needed or desired in the storage system (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 968, 1361-1383; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 249). Bedrock’s expert does not contest that the Prior Art Linux Code meets this limitation (*see* Exh. B, Bedrock’s Rebuttal Expert Report, Attachment 2 at 2-9).

c) The Prior Art Linux Code discloses “a record search means utilizing a search key to access the linked list” in claim 1 and “a record search means utilizing a search key to access a linked list of records having the same hash address” in claim 5.

The Prior Art Linux Code discloses a record search means utilizing a search key to access the linked list of records having the same hash address. The Court, in its Final Claim Construction Order, construed these terms as means-plus-function terms (*see* Claim Construction Order (Dkt. No. 369) at 24-28). The Prior Art Linux Code discloses corresponding structure because it is written to be compiled and stored in memory (RAM) and executed on a computer (CPU).

The hash table disclosed in the Prior Art Linux Code is accessed using a hash key (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1036-1061; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 250). A destination address is used as an input to a hash function, the result of which is the hash key to access the hash table (*see id.*). The resulting hash key is then passed into the rt_cache_add function to access the associated linked list in the hash table (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, l. 1345; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 250). The linked list is then walked in search of a route cache record matching the destination address and, if found, deletes the record from the linked list (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1299-1385; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 250).

```
1036 static void rt_redirect_1(__u32 dst, __u32 gw, struct device *dev)
1037 {
1038        struct rtable *rt;
1039        unsigned long hash = ip_rt_hash_code(dst);
….
1050        rt->rt_dst = dst;
...
1061        rt_cache_add(hash, rt);
```

(*See, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1036-1061 (the function rt_redirect_1 calls the function ip_rt_hash_code to compute a hash key using a hash value called dst, which represents a destination address).) Since a number of possible destination addresses may result in the same hash key, following the hashing computation, all destination addresses that result in the same hash key are found on the same linked list accessed from the hash table. Bedrock's expert does not contest that the Prior Art Linux Code meets this limitation (*see* Exh. B, Bedrock's Rebuttal Expert Report, Attachment 2 at 2-9). Thus, the Prior Art Linux Code discloses these limitations.

d) The Prior Art Linux Code discloses "the record search means including a means for identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" in claims 1 and 5.

The Prior Art Linux Code discloses that the record search means includes a means for identifying and removing[6] at least some of the expired ones of the records from the linked list when the linked list is accessed. The Court construed these terms as means-plus-function terms (*see* Final Claim Construction Order (Dkt. No. 369) at 29-32). The Prior Art Linux Code

[6] Throughout, Defendants use the claim terms in the manner construed by the Court in its Final Claim Construction Order, even though the Defendants disagree with some of those constructions and have objected to them.

discloses corresponding structure because it is written to be compiled and stored in memory (RAM) and executed on a computer (CPU).

The Prior Art Linux Code discloses that during the access to the linked list (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1372, 1378) to search for a duplicate record to delete, the Prior Art Linux Code both *identifies* and *removes* route cache records that have expired (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1361-1383; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 252). Both identification and removal of route cache records occur during the same access to the linked list. Bedrock' expert does not contest that identification and removal of route cache records occur during the same access as the search for a duplicate record to delete. (*see* Exh. B, Bedrock's Rebuttal Expert Report, Attachment 2 at 2-4, 6-7).

```
1299 static void rt_cache_add(unsigned hash, struct rtable * rth)
1300 {
1301        unsigned long flags;
1302        struct rtable **rthp;
1303        __u32 daddr = rth->rt_dst;
…
1341        if (rt_cache_size >= RT_CACHE_SIZE_MAX)
1342                rt_garbage_collect();
1343
1344        cli();
1345        rth->rt_next = ip_rt_hash_table[hash];
1346        #if RT_CACHE_DEBUG >= 2
1347                if (rth->rt_next)
1348                {
1349                        struct rtable * trth;
1350                        printk("rt_cache @%02x: %08x", hash, daddr);
1351                        for (trth=rth->rt_next; trth; trth=trth->rt_next)
1352                                printk(" . %08x", trth->rt_dst);
1353                        printk("\n");
1354                }
1355        #endif
1356        ip_rt_hash_table[hash] = rth;
1357        rthp = &rth->rt_next;
1358        sti();
1359        rt_cache_size++;
1360

1361        /*
1362        * Cleanup duplicate (and aged off) entries.
1363        */
1364
1365        while ((rth = *rthp) != NULL)
1366        {
1367
1368                cli();
1369            if ((!rth->rt_refcnt && rth->rt_lastuse + RT_CACHE_TIMEOUT < now)
1370            || rth->rt_dst == daddr)
1371            {
1372                    *rthp = rth->rt_next;
1373                    rt_cache_size--;
1374                    sti();
1375 #if RT_CACHE_DEBUG >= 2
1376                    printk("rt_cache clean %02x@%08x\n", hash, rth->rt_dst);
1377 #endif
1378                    rt_free(rth);
1379                    continue;
1380            }
1381            sti();
1382            rthp = &rth->rt_next;
1383    }

1384    restore_flags(flags);
1385 }
```

(Exh. A, Linux 2.0.1 at route.c, ll. 1299-1385.)

As discussed above, and based on Bedrock's interpretation of the court's construction of "expiration," the Prior Art Linux Code identifies an expired record based on whether the reference count is zero and the lifetime of the record has passed (*see, e.g.,* Exh. A, Linux 2.0.1 at route.c, ll. 1369-1370; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 253; Exh. H, Bedrock's Infringement Contention at 1-4; Bedrock's Opening Expert Report at 40-41). After an expired record is identified, it is then removed from the linked list by adjusting the pointers in the linked list to bypass the record (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 254).

e) The Prior Art Linux Code discloses a "means, utilizing the record search means, for accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list" in claim 1 and a "means, utilizing the record search means, for inserting, retrieving, and deleting records from the system and, at the same time, removing at least some expired ones of the records in the accessed inked list of records" in claim 5.

The Prior Art Linux Code discloses a means, utilizing the record search means, for accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list. The Court construed these terms as means-plus-function terms (*see* Final Claim Construction Order (Dkt. No. 369) at 33-36, 42-43). The Prior Art Linux Code discloses corresponding structure because it is written to be compiled and stored in memory (RAM) and executed on a computer (CPU).

The Prior Art Linux Code discloses that the rt_cache_add function is called by the ip_rt_slow_route and rt_redirect_1 functions, which use rt_cache_add to insert a record into the hash table, retrieve a record from the table to print, and delete any duplicate records from the hash table, and at the same time, remove expired records from the hash table (*see, e.g.,* Exh. A, Linux 2.0.1 at route.c, ll. 1036, 1299-1385; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 255).

Further, the function rt_cache_add inserts, retrieves, and deletes records from the hash table at the same time that it removes at least some of the expired ones of the records from the accessed linked list of records (*see, e.g.,* Exh. A, Linux 2.0.1 at route.c, ll. 1299-1385; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 256.) First, rt_cache_add *inserts* a record into the linked list at lines 1356-57. Second, after the insertion of the record into the linked list, the rt_cache_add function traverses the remainder of the linked list in search of the record just added so that it can *delete* any duplicate records to the record just added (*see* Exh. E, Jeffay Decl. Exh. A at ¶¶ 256-258). During this traversal in search of duplicate records to delete, any expired records are identified and removed (*see, e.g*., Exh. A, Linux 2.0.1 at route.c, lines 1361-1383; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 258). Third, when a duplicate record or an expired record is found, the rt_cache_add function then *retrieves* the record that was just removed so that it can print it using the printk function at line 1376 (*see, e.g*., Exh. A, Linux 2.0.1 at route.c, l. 1376; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 259).

If Bedrock interprets this claim element to include the deletion of a record in a separate traversal of the linked list that occurs prior to the removal of expired records (*see, e.g.,* Exh. H, Bedrock's Infringement Contention at 1-4), the Prior Art Linux Code also discloses a second way in which the rt_cache_add function is used to insert a record into the hash table, retrieve a record from the table to print, and delete any duplicate records from the hash table, and at the same time, remove expired records from the hash table. The insertion described above and the deletion of records during a call to function rt_garbage_collect at line 1342 discloses this element (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1342, 1356, 1289-1297, 1107-1137; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 260).

```
1289    static __inline__ void rt_garbage_collect(void)
1290    {
1291            if (ip_rt_lock == 1)
1292            {
1293                    rt_garbage_collect_1();
1294                    return;
1295            }
1296            ip_rt_bh_mask |= RT_BH_GARBAGE_COLLECT;
1297    }

1107    static void rt_garbage_collect_1(void)
1108    {
1109    int i;
1110    unsigned expire = RT_CACHE_TIMEOUT>>1;
1111    struct rtable * rth, **rthp;
1112    unsigned long now = jiffies;
1113
1114    for (;;)
1115    {
1116            for (i=0; i<RT_HASH_DIVISOR; i++)

1117    {
1118            if (!ip_rt_hash_table[i])
1119                    continue;
1120            for (rthp=&ip_rt_hash_table[i]; (rth=*rthp); rthp=&rth->rt_next)
1121            {
1122                    if (rth->rt_lastuse + expire*(rth->rt_refcnt+1) > now)
1123                            continue;
1124                    rt_cache_size--;
1125                    cli();
1126                    *rthp=rth->rt_next;
1127                    rth->rt_next = NULL;
1128                    sti();
1129                    rt_free(rth);
1130                    break;
1131            }
1132    }
1133    if (rt_cache_size < RT_CACHE_SIZE_MAX)
1134            return;
1135    expire >>= 1;
1136    }
1137    }
```

(Exh. A, Linux 2.0.1 at route.c, ll. 1289-1297, 1107-1137.) The function rt_garbage_collect calls the function rt_garbage collect_1, which causes the function to sequentially step through each linked list of records in the hash table (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, l. 1116). The function then accesses a linked list at line 1120 and examines each record to identify expired records at line 1122. If the record has expired, the record is removed from the list at lines 1126-27 and the record is deallocated at line 1129 by function rt_free, which causes the record to be returned to the operating system's free memory pool (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 260). Bedrock's expert does not contest that the Prior Art Linux Code meets this limitation (*see* Exh. B, Bedrock's Rebuttal Expert Report, Attachment 2 at 2-9).

The Prior Art Linux code teaches every element of claims 1 and 5 of the '120 Patent. Therefore, there are no genuine issues of material fact that the Prior Art Linux Code anticipates claims 1 and 5 of the '120 Patent.

2. The Prior Art Linux Code Anticipates Claims 2 and 6 of the '120 Patent

The Linux Prior Art Code discloses a means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records as set forth in dependent claims 2 and 6 of the '120 Patent. The Court, in its Final Claim Construction Order, construed this term as a means-plus-function term (*see* Final Claim Construction Order (Dkt. No. 369 at 39-41). The Prior Art Linux Code discloses corresponding structure because it is written to be compiled and stored in memory (RAM) and executed on a computer (CPU).

Under Bedrock's interpretation of the court's construction of this claim term, a determination of whether or not to remove one expired record falls within the meaning of this claim term (*see, e.g.*, Exh. H, Bedrock's Infringement Contention at 1-4; *see also* Bedrock's Opening Expert Report at 40-41). The Prior Art Linux Code contains a conditional statement that determines whether or not to remove an expired record by comparing the record's expiration time with the current time (*see, e.g.*, Exh. A, Linux 2.0.1 at route.c, ll. 1369-80; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 263).

```
1369            if ((!rth->rt_refcnt && rth->rt_lastuse + RT_CACHE_TIMEOUT < now)
1370            || rth->rt_dst == daddr)
1371            {
1372                    *rthp = rth->rt_next;
1373                    rt_cache_size--;
1374                    sti();
1375 #if RT_CACHE_DEBUG >= 2
1376                    printk("rt_cache clean %02x@%08x\n", hash, rth->rt_dst);
1377 #endif
1378                    rt_free(rth);
1379                    continue;
1380            }
```

(Exh. A, Linux 2.0.1 at route.c, ll. 1369-80.) This determination of whether or not to remove an expired record falls squarely within Bedrock's interpretation of the court's construction of this claim term (*see* Exh. E, Jeffay Decl. Exh. A at ¶ 263).

The Prior Art Linux Code also discloses a second way for dynamically determining maximum number for the record search means to remove in the accessed linked list of records. When rt_cache_add is called to insert a route cache record, if the number of records in the hash table is greater than the predetermined "threshold" value, a call to rt_garbage_collect is made, which in turn calls rt_garbage_collect_1. This function proceeds to remove expired records from the hash table until the hash table contains a number of route cache records that is less than the predetermined "threshold" value, at which point the function stops removing records (*see* Exh. A, Linux 2.0.1 at route.c, ll. 1107-1116, 1289-1297, 1341-1342). This threshold value dynamically determines the number of records to remove from the linked list during removal procedure set forth in rt_garbage_collect_1.

The Prior Art Linux code teaches every element of claims 2 and 6 of the '120 Patent. Therefore, there are no genuine issues of material fact that the Prior Art Linux Code anticipates claims 2 and 6 of the '120 Patent.

3. The Prior Art Linux Code Anticipates Claims 3 and 7 of the '120 Patent

The Prior Art Linux Code discloses each step of method claims 3 and 7. As previously discussed, the Prior Art Linux Code discloses a method for storing and retrieving information records using a linked list and/or a hash table with external chaining to store and provide access to records, some of which automatically expire based on whether the record's reference count is zero and whether the record's lifetime has passed (*see supra* Sections I.A.1.a – I.A.1.e; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 267).

The Prior Art Linux code discloses accessing a linked list of records having the same hash address in the rt_cache_add function (*see supra* Sections I.A.1.a - I.A.1.c; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 268). The Prior Art Linux Code discloses identifying and removing at least some of the automatically expired ones of the records when the linked list is accessed in the rt_cache_add function, where the function identifies and removes automatically expired records while the linked list is being accessed in search of a duplicate record to the one just inserted (*see supra* Section2 I.A.1.c - I.A.1.d; *see also* Exh. E, Jeffay Decl. Exh. A at ¶ 270). Finally, the Prior Art Linux Code also discloses inserting, retrieving, or deleting one of the records from the system following the step of removing in rt_cache_add, where the function searches for and removes duplicate records and expired records. This deletion may occur before or after expired records have been identified (*see* Supra Section I.A.1.d; *see also* Exh. E, Jeffay Decl. Exh. A at ¶¶ 272-274).

The Prior Art Linux code teaches every element of claims 3 and 7 of the '120 Patent. In fact, Bedrock's expert does not even attempt to rebut the Defendants' experts' arguments that the Prior Art Linux code anticipates these claims (*see* Exh. B, Bedrock's Rebuttal Expert Report, Attachment 2 at 2-9). Therefore, there are no genuine issues of material fact that the Prior Art Linux Code anticipates claims 3 and 7 of the '120 Patent.

4. The Prior Art Linux Code Anticipates Claims 4 and 8 of the ’120 Patent

As previously discussed, the Prior Art Linux Code discloses the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed (*see* Supra Section I.A.2; *see also* Exh. E, Jeffay Decl. Exh. A at ¶¶ 275-277).

The Prior Art Linux code teaches every element of claims 4 and 8 of the ’120 Patent. In fact, Bedrock’s expert does not even attempt to rebut the Defendants’ experts’ arguments that the Prior Art Linux code anticipates these claims (*see* Exh. B, Bedrock’s Rebuttal Expert Report, Attachment 2 at 2-9). Therefore, there are no genuine issues of material fact that the Prior Art Linux Code anticipates claims 4 and 8 of the ’120 Patent.

VI. CONCLUSION

For the reasons explained above, Defendants respectfully request that the Court grant Defendants' motion for summary judgment that claims 1-8 of the '120 Patent are invalid as anticipated and/or obvious.

Respectfully submitted, this the 8th day of February 2011.

/s/ E. Danielle T. Williams (with permission)
Steven Gardner
E. Danielle T. Williams
John C. Alemanni
KILPATRICK TOWNSEND &
STOCKTON LLP
1001 West 4th Street
Winston-Salem, NC 27101
Telephone: 336-607-7300
Fax: 336-607-7500

William H. Boice
Russell A. Korn
KILPATRICK TOWNSEND &
STOCKTON LLP
Suite 2800
1100 Peachtree Street
Atlanta, GA 30309-4530
Telephone: 404-815-6500
Fax: 404-815-6555

J. Thad Heartfield
Texas Bar No. 09346800
thad@jth-law.com
M. Dru Montgomery
Texas Bar No. 24010800
dru@jth-law.com
THE HEARTFIELD LAW FIRM
2195 Dowlen Road
Beaumont, TX 77706
Telephone: 409-866-2800
Fax: 409-866-5789

Attorneys for Defendants Softlayer Technologies, Inc. and Amazon.com, Inc.

/s/ Deron R. Dacus
Deron R. Dacus
Texas Bar No. 00790553
derond@rameyflock.com
Ramey & Flock, P.C.
100 E. Ferguson, Suite 500
Tyler, Texas 75702
Telephone: (903) 597-3301
Facsimile: (903) 597-2413

Frank G. Smith
frank.smith@alston.com
ALSTON & BIRD LLP
One Atlantic Center
1201 West Peachtree Street
Atlanta, GA 30309
Telephone: (404) 881-7240
Facsimile: (404) 256-8184

Alan L. Whitehurst
alan.whitehurst@alston.com
Marissa R. Ducca
marissa.ducca@alston.com
ALSTON & BIRD LLP
The Atlantic Building
950 F Street, N.W.
Washington, DC 20004
Telephone: (202) 756-3300
Facsimile: (202) 756-3333

Michael J. Newton (SBN 24003844)
mike.newton@alston.com
ALSTON & BIRD LLP
Chase Tower
2200 Ross Avenue, Suite 3601
Dallas, TX 75201
Telephone: (214) 922-3423
Facsimile: (214) 922-3839

Louis A. Karasik (*pro hac vice*)
lou.karasik@alston.com
Rachel Capoccia
rachel.capoccia@alston.com
ALSTON & BIRD LLP

333 South Hope Street
16th Floor
Los Angeles, CA 90071
Telephone: (213) 576-1148
Facsimile: (213) 576-1100

Attorneys for Defendants AOL Inc. and Myspace, Inc.

/s/ John A. Lee (with permission)
Yar R. Chaikovsky
California State Bar No. 175421
John A. Lee
California State Bar No. 229911
McDERMOTT WILL & EMERY LLP
275 Middlefield Road, Suite 100
Menlo Park, CA 94025
Tel: 650.815.7400
Fax: 650.815.7401
E-mail: ychaikovsky@mwe.com
Email: jlee@mwe.com

Christopher D. Bright
Cal. Bar No. 206273
McDERMOTT WILL & EMERY LLP
18191 Von Karman Ave, Ste. 500
Irvine, California 92612
Tel: 949.757.7178
Fax: 949.851.9348
E-mail: cbright@mwe.com

Attorneys for Defendant Yahoo! Inc.

/s/ Todd Briggs (with permission)
Claude M. Stern
claudestern@quinnemanuel.com
Evette D. Pennypacker
evettepennypacker@quinnemanuel.com
Todd M. Briggs
toddbriggs@quinnemanuel.com
Antonio Sistos
antoniosistos@quinnemanuel.com
QUINN EMANUEL URQUHART & SULLIVAN, LLP
555 Twin Dolphin Dr., 5th Floor
Redwood Shores, CA 94065
Telephone: 650-801-5000
Facsimile: 650-801-5100

Michael E. Jones
mikejones@potterminton.com
Texas State Bar No. 10929400
POTTER MINTON, PC
110 N. College
Tyler, Texas 75702
Telephone: (903) 597-8311
Facsimile: (903) 593-0846

Attorneys for Google, Inc. and Match.com, LLC

CERTIFICATE OF SERVICE

This is to certify that all counsel of record who are deemed to have consented to electronic service are being served with a copy of this document via the Court's CM/ECF system per Local Rule CV-5(a)(3) on this 8th day of February, 2011. Any other counsel of record will be served by first class mail.

______/s/ Deron R. Dacus_____