EXHIBIT 8

	Claim Language	**Court's Construction**	**Accused Instrumentalities:** **Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11**
1.	An information storage and retrieval system, the system comprising:		Bedrock Computer Technologies LLC ("Bedrock") does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted for information storage and retrieval.
(a)[1]	**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**,	**a linked list to store and provide access to records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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. Within Linux kernel version 2.6.11, the data structure rt_hash_table in module /net/ipv4/route.c[2] anchors one or more linked list(s) to store and

[1] While the limitations are not lettered in the actual claims of the patent, Bedrock provides them here for ease of reference.

[2] The path names of the cited source code is provided for the defendants' convenience. If any version or customization of Linux kernel version 2.6.11 deviates from the path names that are cited in these charts, such deviations are insignificant because it is the routines, functions, methods, macros, classes, data structures, etc., as embodied on servers and other devices, that infringe.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			provide access to records stored in a memory of the system, at least some of the records automatically expiring. In this way, computer equipment configured with or utilizing software based on 2.6.11 includes 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. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `  spinlock_t lock;` `} __attribute__((__aligned__(8)));` `struct rtable` `{` `  union` `  {` `  struct dst_entry dst;` `  struct rtable *rt_next;` `  } u;` `  struct in_device *idev;` `  unsigned rt_flags;` `  unsigned rt_type;` `  __u32 rt_dst; /* Path destination */` `  __u32 rt_src; /* Path source */` `  int rt_iif;` `  /* Info on neighbour */` `  __u32 rt_gateway;` `  /* Cache lookup keys */` `  struct flowi fl;` `  /* Miscellaneous cached information */`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			`__u32 rt_spec_dst; /* RFC1122 specific destination */` `struct inet_peer *peer; /* long-living peer info */` `};` Source: Linux kernel source code files /net/ipv4/route.c and /include/net/route.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system.
(b)	**a record search means utilizing a search key to access the linked list,**	**function:** "utilizing a search key to access the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 31-36 and Boxes 39-41 of FIG. 3 and in col. 5 line 53-col. 6 line 4 and col. 6 lines 14-20, and/or	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means utilizing a search key to access the linked list or its equivalent. The following code within route.c performs the function of utilizing a

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
		programmed with software instructions as described in the pseudocode of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	search key to access the linked list: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5), tos);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5), tos);` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5), tos);` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5), tos);` `err = rt_intern_hash(hash, rth, rp);` `static unsigned int rt_hash_code(u32 daddr, u32 saddr, u8 tos)` `{` `return (jhash_3words(daddr, saddr, (u32) tos, rt_hash_rnd)` `& rt_hash_mask);` `}` rt_intern_hash accesses the linked list and searches for a record by comparing keys: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(&rt_hash_table[hash].lock);` `while ((rth = *rthp) != NULL) {` `if (compare_keys(&rth->fl, &rt->fl)) {` **** `*rp = rth;` `return 0;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			} **** chain_length++; rthp = &rth->u.rt_next; } **** spin_unlock_bh(&rt_hash_table[hash].lock); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 && fl1->oif == fl2->oif && fl1->iif == fl2->iif; } Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(c)	**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**, and	**function:** "identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means, 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 or its equivalent.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
		described in Boxes 33-42 of FIG. 3 and in col. 5 line 53-col. 6 line 34, and/or programmed with software instructions as described in the pseudo-code of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	Specifically, code contained within function rt_intern_hash comprises 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 or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed: `spin_lock_bh(&rt_hash_table[hash].lock);` `while ((rth = *rthp) != NULL) {` `if (compare_keys(&rth->fl, &rt->fl)) {` `****` `*rp = rth;` `return 0;` `}` `****` `if (!atomic_read(&rth->u.dst.__refcnt)) {` `u32 score = rt_score(rth);` `if (score <= min_score) {` `cand = rth;` `candp = rthp;` `min_score = score;` `}` `}` `chain_length++;` `rthp = &rth->u.rt_next;` `}` `if (cand) {` `/* ip_rt_gc_elasticity used to be average length of chain` `* length, when exceeded gc becomes really aggressive.` `*` `* The second limit is less certain. At the moment it allows` `* only 2 entries per bucket. We will see.` `*/` `if (chain_length > ip_rt_gc_elasticity) {` `*candp = cand->u.rt_next;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			rt_free(cand); } } **** spin_unlock_bh(&rt_hash_table[hash].lock); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 && fl1->oif == fl2->oif && fl1->iif == fl2->iif; } static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c Note that the record(s) identified as expired upon traversal of the linked list

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			is not necessarily the record that rt_intern_hash was called to find. Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash. Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(d)	**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**.	**function:** "utilizing the record search means, accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions that provide the insert, retrieve, or delete record capability as described in the flowchart of FIG. 5 and col. 7 line 65 – col. 8 line 32, FIG. 6 and col. 8 lines 33-44, or FIG. 7 and col. 8 lines 45-59, respectively, and/or programmed with software instructions that provide the insert, retrieve or delete record capability as described in the pseudocode of Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9, 10, 11, and 12), or Delete Procedure (cols. 11 and 12), respectively, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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 or its equivalent. The code identified below collectively performs the function of utilizing the record search means, accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list. This function is parsed out below for convenience. The following calls to the hashing function and rt_intern_hash are all utilizations of the record search means: `unsigned hash = rt_hash_code(daddr,`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			skeys[i] ^ (ikeys[k] << 5), tos); if (!rt_intern_hash(hash, rt, &rt)) * or * hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5), tos); return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst); * or * hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5), tos); err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst); * or * hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5), tos); err = rt_intern_hash(hash, rth, rp); static unsigned int rt_hash_code(u32 daddr, u32 saddr, u8 tos) { return (jhash_3words(daddr, saddr, (u32) tos, rt_hash_rnd) & rt_hash_mask); } rt_intern_hash, inserts a record: static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp) { struct rtable *rth, **rthp; unsigned long now; struct rtable *cand, **candp; u32 min_score; int chain_length; int attempts = !in_softirq(); **** rt->u.rt_next = rt_hash_table[hash].chain; #if RT_CACHE_DEBUG >= 2 if (rt->u.rt_next) { struct rtable *trt; printk(KERN_DEBUG "rt_cache @%02x: %u.%u.%u.%u", hash, NIPQUAD(rt->rt_dst));

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			for (trt = rt->u.rt_next; trt; trt = trt->u.rt_next) printk(" . %u.%u.%u.%u", NIPQUAD(trt->rt_dst)); printk("\n"); } #endif rt_hash_table[hash].chain = rt; spin_unlock_bh(&rt_hash_table[hash].lock); *rp = rt; return 0; } rt_intern_hash retrieves a record: static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp) { struct rtable *rth, **rthp; unsigned long now; struct rtable *cand, **candp; u32 min_score; int chain_length; int attempts = !in_softirq(); **** /* Put it first */ *rthp = rth->u.rt_next; /* * Since lookup is lockfree, the deletion * must be visible to another weakly ordered CPU before * the insertion at the start of the hash chain. */ rcu_assign_pointer(rth->u.rt_next, rt_hash_table[hash].chain); /* * Since lookup is lockfree, the update writes * must be ordered for consistency on SMP. */ rcu_assign_pointer(rt_hash_table[hash].chain, rth); rth->u.dst.__use++; dst_hold(&rth->u.dst); rth->u.dst.lastuse = now; spin_unlock_bh(&rt_hash_table[hash].lock);

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			rt_drop(rt); *rp = rth; return 0; } rt_intern_hash is also invoked when deleting a record: unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5), tos); rthp=&rt_hash_table[hash].chain; rcu_read_lock(); while ((rth = rcu_dereference(*rthp)) != NULL) { struct rtable *rt; if (rth->fl.fl4_dst != daddr || rth->fl.fl4_src != skeys[i] || rth->fl.fl4_tos != tos || rth->fl.oif != ikeys[k] || rth->fl.iif != 0) { rthp = &rth->u.rt_next; continue; } if (rth->rt_dst != daddr || rth->rt_src != saddr || rth->u.dst.error || rth->rt_gateway != old_gw || rth->u.dst.dev != dev) break; dst_hold(&rth->u.dst); rcu_read_unlock(); rt = dst_alloc(&ipv4_dst_ops); if (rt == NULL) { ip_rt_put(rth); in_dev_put(in_dev); return; } ****

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			rt_del(hash, rth); if (!rt_intern_hash(hash, rt, &rt)) ip_rt_put(rt); goto do_next; } Source: Linux kernel source code file /net/ipv4/route.c As explained above, code within rt_intern_hash performs the function of accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list. Bedrock contends that that this limitation is literally met both identically and by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
2.	The information storage and retrieval system according to claim 1 further including **means for dynamically determining maximum number** for the record search means to remove in the accessed linked list of records.	**function:** "dynamically determining maximum number for the record search means to remove in the accessed linked list of records" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to dynamically determine a maximum number of records to remove by choosing a search strategy of removing all expired records from a linked list or removing some but not all of the expired records as described in col. 6 line 56 – col. 7 line 15 and/or programmed with software instructions to dynamically determine a maximum number of records to remove by	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 includes means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
		choosing between the pseudocode of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	The following code excerpt from the rt_intern_hash function performs the function of dynamically determining maximum number for the record search means to remove in the accessed linked list of records: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
3.	A method for storing and retrieving information records using a **linked list to store and provide access to the records**, at least some of the records **automatically expiring**, the method comprising the steps of:	**a linked list to store and provide access to the records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	Bedrock does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method for storing and retrieving information records that uses a linked list to store and provide access to the records, where at least some of the records are automatically expiring. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to store and retrieve information records using a linked list to store and provide access to the records, where at least some of the records are automatically expiring. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `  spinlock_t lock;` `} __attribute__((__aligned__(8)));` `struct rtable` `{` `  union` `  {` `  struct dst_entry dst;` `  struct rtable *rt_next;` `  } u;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			struct in_device *idev; unsigned rt_flags; unsigned rt_type; __u32 rt_dst; /* Path destination */ __u32 rt_src; /* Path source */ int rt_iif; /* Info on neighbour */ __u32 rt_gateway; /* Cache lookup keys */ struct flowi fl; /* Miscellaneous cached information */ __u32 rt_spec_dst; /* RFC1122 specific destination */ struct inet_peer *peer; /* long-living peer info */ }; Source: Linux kernel source code files /net/ipv4/route.c and /include/net/route.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
(a)	accessing the **linked list** of records,	**linked list** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of accessing the linked list of records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to access a linked list of records. Specifically, the data structure rt_hash_table in module /net/ipv4/route.c is used to access the linked list of records. Additionally, code contained within the function rt_intern_hash in module /net/ipv4/route.c is also used to access the linked list of records. The following code excerpts within route.c performs the step of accessing a linked list of records: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5), tos);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5), tos);` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5), tos);` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5), tos);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			`err = rt_intern_hash(hash, rth, rp);` `static unsigned int rt_hash_code(u32 daddr, u32 saddr, u8 tos)` `{` `return (jhash_3words(daddr, saddr, (u32) tos, rt_hash_rnd)` `& rt_hash_mask);` `}` rt_intern_hash accesses the linked list: `rthp = &rt_hash_table[hash].chain;` Source: Linux kernel source code file /net/ipv4/route.c
(b)	identifying at least some of the automatically **expired** ones of the records, and	**expired** means "obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of identifying at least some of the automatically expired ones of the records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to identify at least some of the automatically expired ones of the records. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. Code contained within or accessed by the function rt_intern_hash in module /net/ipv4/route.c uses rt_score to practice a method that includes the step of

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			identifying at least some of the automatically expired ones of the records. The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: `spin_lock_bh(&rt_hash_table[hash].lock);` `while ((rth = *rthp) != NULL) {` `if (compare_keys(&rth->fl, &rt->fl)) {` `****` `*rp = rth;` `return 0;` `}` `****` `if (!atomic_read(&rth->u.dst.__refcnt)) {` `u32 score = rt_score(rth);` `if (score <= min_score) {` `cand = rth;` `candp = rthp;` `min_score = score;` `}` `}` `chain_length++;` `rthp = &rth->u.rt_next;` `}` `if (cand) {` Source: Linux kernel source code file /net/ipv4/route.c
(c)	**removing at least some of the automatically expired records from the linked list when the linked list is accessed**.	**removing at least some of the automatically expired records from the linked list** means "adjusting the pointer in the linked list to bypass the previously identified expired records" **when the linked list is accessed** means "both identification	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. Computer equipment configured with or utilizing software based

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
		and removal of the automatically expired record(s) occurs during the same access of the linked list"	on Linux kernel version 2.6.11 is especially adapted to remove at least some of the automatically expired records from the linked list when the linked list is accessed. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the linked list when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` The line of code "*candp = cand->u.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records. Source: Linux kernel source code file /net/ipv4/route.c Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			rt_intern_hash.
	{ordering of the steps}	**ordering of the steps:** The "identifying" step must start before "removal" can begin. However, identification need not be completed before removal can begin. The identification step may overlap with the removal step.	As shown in the code below, the "identifying" step starts before the "removal" step: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}`
4.	The method according to claim 3 further including the step of **dynamically determining** maximum number of expired ones of the records to remove when the linked list is accessed.	**dynamically determining** means "making a decision based on factors internal or external to the information storage and retrieval system"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Specifically, code contained within function rt_intern_hash (in module /net/ipv4/route.c) that dynamically executes based upon comparison with variable ip_rt_gc_elasticity is used to perform the claimed act(s). In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 practices a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to dynamically determine maximum number of expired ones of the

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			records to remove when the linked list is accessed. The following code excerpt from the rt_intern_hash function performs the step of dynamically determining the maximum number of expired ones of the records to remove when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;` The line of code "if (chain_length > ip_rt_gc_elasticity)" therefore makes a decision based on factors internal or external to the information storage and retrieval system. Source: Linux kernel source code file /net/ipv4/route.c
5.	An information storage and retrieval system, the system comprising:		Bedrock Computer Technologies LLC ("Bedrock") does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			Accused Instrumentalities that correspond to the claim preamble. When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted for information storage and retrieval.
(a)	**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**,	**function:** "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" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to provide a hash table having a pointer to the head of a linked list of externally chained records as described in col. 5 lines 16-26 and/or programmed with software instructions as described in the pseudo-code of Definitions, definition number 4, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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, where at least some of the records are automatically expiring or its equivalent. Specifically, data structure rt_hash_table in module /net/ipv4/route.c implements 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, where at least some of the records automatically are expiring or its equivalent. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			struct rt_hash_bucket { struct rtable *chain; spinlock_t lock; } __attribute__((__aligned__(8))); struct rtable { union { struct dst_entry dst; struct rtable *rt_next; } u; struct in_device *idev; unsigned rt_flags; unsigned rt_type; __u32 rt_dst; /* Path destination */ __u32 rt_src; /* Path source */ int rt_iif; /* Info on neighbour */ __u32 rt_gateway; /* Cache lookup keys */ struct flowi fl; /* Miscellaneous cached information */ __u32 rt_spec_dst; /* RFC1122 specific destination */ struct inet_peer *peer; /* long-living peer info */ }; ipv4_dst_ops.kmem_cachep = kmem_cache_create("ip_dst_cache", sizeof(struct rtable), 0, SLAB_HWCACHE_ALIGN, NULL, NULL); **** goal = num_physpages >> (26 - PAGE_SHIFT); if (rhash_entries) goal = (rhash_entries * sizeof(struct rt_hash_bucket)) >> PAGE_SHIFT; for (order = 0; (1UL << order) < goal; order++) /* NOTHING */;

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			`do {` `rt_hash_mask = (1UL << order) * PAGE_SIZE /` `sizeof(struct rt_hash_bucket);` `while (rt_hash_mask & (rt_hash_mask - 1))` `rt_hash_mask--;` `rt_hash_table = (struct rt_hash_bucket *)` `__get_free_pages(GFP_ATOMIC, order);` `} while (rt_hash_table == NULL && --order > 0);` `****` `rt_hash_mask--;` `for (i = 0; i <= rt_hash_mask; i++) {` `spin_lock_init(&rt_hash_table[i].lock);` `rt_hash_table[i].chain = NULL;` Source: Linux kernel source code file /net/ipv4/route.c
(b)	**a record search means utilizing a search key to access a linked list of records having the same hash address**,	**function:** "utilizing a search key to access the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 31-36 and Boxes 39-41 of FIG. 3 and in col. 5 line 53-col. 6 line 4 and col. 6 lines 14-20, and/or programmed with software instructions as described in the pseudocode of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means utilizing a search key to access a linked list of records having the same hash address or its equivalent. The following code within route.c performs the function of utilizing a search key to access the linked list: `unsigned hash = rt_hash_code(daddr,` `skeys[i] ^ (ikeys[k] << 5),` `tos);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5), tos);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			`return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5), tos);` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5), tos);` `err = rt_intern_hash(hash, rth, rp);` `static unsigned int rt_hash_code(u32 daddr, u32 saddr, u8 tos)` `{` `return (jhash_3words(daddr, saddr, (u32) tos, rt_hash_rnd)` `& rt_hash_mask);` `}` rt_intern_hash accesses the linked list and searches for a record by comparing keys: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(&rt_hash_table[hash].lock);` `while ((rth = *rthp) != NULL) {` `if (compare_keys(&rth->fl, &rt->fl)) {` `****` `*rp = rth;` `return 0;` `}` `****` `chain_length++;` `rthp = &rth->u.rt_next;` `}` `****` `spin_unlock_bh(&rt_hash_table[hash].lock);` `*rp = rt;` `return 0;` `}` `static inline int compare_keys(struct flowi *fl1, struct flowi *fl2)` `{`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			`return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 &&` `fl1->oif == fl2->oif &&` `fl1->iif == fl2->iif;` `}` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(c)	**the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed**, and	**function:** "identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 33-42 of FIG. 3 and in col. 5 line 53-col. 6 line 34, and/or programmed with software instructions as described in the pseudo-code of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed or its equivalent. Specifically, code contained within function rt_intern_hash comprises 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 or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			spin_lock_bh(&rt_hash_table[hash].lock); while ((rth = *rthp) != NULL) { if (compare_keys(&rth->fl, &rt->fl)) { **** *rp = rth; return 0; } **** if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score; } } chain_length++; rthp = &rth->u.rt_next; } if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.rt_next; rt_free(cand); } } **** spin_unlock_bh(&rt_hash_table[hash].lock); *rp = rt; return 0; } static inline int compare keys(struct flowi *fl1, struct flowi *fl2)

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			{ return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 && fl1->oif == fl2->oif && fl1->iif == fl2->iif; } static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c Note that the record(s) identified as expired upon traversal of the linked list is not necessarily the record that rt_intern_hash was called to find. Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash. Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(d)	**mea[n]s, 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 linked list of records**.	**function:** "utilizing the record search means, 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 linked list of records" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions that provide the insert, retrieve, and delete record capability as described in the flowchart of FIG. 5 and col. 7 line 65 – col. 8 line 32, FIG. 6 and col. 8 lines 33-44, or FIG. 7 and col. 8 lines 45-59, respectively, and/or programmed with software instructions that provide the insert, retrieve and delete record capability as described in the pseudo-code of Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9, 10, 11, and 12), and Delete Procedure (cols. 11 and 12), respectively, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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 linked list of records or its equivalent. The code identified below collectively performs the function of utilizing the record search means, 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 linked list of records. This function is parsed out below for convenience. Specifically, the following calls to the hashing function and rt_intern_hash are all utilizations of the record search means: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5), tos);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5), tos);` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			* or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5), tos);` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5), tos);` `err = rt_intern_hash(hash, rth, rp);` `static unsigned int rt_hash_code(u32 daddr, u32 saddr, u8 tos)` `{` `return (jhash_3words(daddr, saddr, (u32) tos, rt_hash_rnd)` `& rt_hash_mask);` `}` rt_intern_hash, inserts a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)` `{` `struct rtable *rth, **rthp;` `unsigned long now;` `struct rtable *cand, **candp;` `u32 min_score;` `int chain_length;` `int attempts = !in_softirq();` `****` `rt->u.rt_next = rt_hash_table[hash].chain;` `#if RT_CACHE_DEBUG >= 2` `if (rt->u.rt_next) {` `struct rtable *trt;` `printk(KERN_DEBUG "rt_cache @%02x: %u.%u.%u.%u", hash,` `NIPQUAD(rt->rt_dst));` `for (trt = rt->u.rt_next; trt; trt = trt->u.rt_next)` `printk(" . %u.%u.%u.%u", NIPQUAD(trt->rt_dst));` `printk("\n");` `}` `#endif` `rt_hash_table[hash].chain = rt;` `spin_unlock_bh(&rt_hash_table[hash].lock);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			*rp = rt; return 0; } rt_intern_hash retrieves a record: static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp) { struct rtable *rth, **rthp; unsigned long now; struct rtable *cand, **candp; u32 min_score; int chain_length; int attempts = !in_softirq(); **** /* Put it first */ *rthp = rth->u.rt_next; /* * Since lookup is lockfree, the deletion * must be visible to another weakly ordered CPU before * the insertion at the start of the hash chain. */ rcu_assign_pointer(rth->u.rt_next, rt_hash_table[hash].chain); /* * Since lookup is lockfree, the update writes * must be ordered for consistency on SMP. */ rcu_assign_pointer(rt_hash_table[hash].chain, rth); rth->u.dst.__use++; dst_hold(&rth->u.dst); rth->u.dst.lastuse = now; spin_unlock_bh(&rt_hash_table[hash].lock); rt_drop(rt); *rp = rth; return 0; }

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			rt_intern_hash is also invoked when deleting a record: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5), tos);` `rthp=&rt_hash_table[hash].chain;` `rcu_read_lock();` `while ((rth = rcu_dereference(*rthp)) != NULL) {` `struct rtable *rt;` `if (rth->fl.fl4_dst != daddr ||` `rth->fl.fl4_src != skeys[i] ||` `rth->fl.fl4_tos != tos ||` `rth->fl.oif != ikeys[k] ||` `rth->fl.iif != 0) {` `rthp = &rth->u.rt_next;` `continue;` `}` `if (rth->rt_dst != daddr ||` `rth->rt_src != saddr ||` `rth->u.dst.error ||` `rth->rt_gateway != old_gw ||` `rth->u.dst.dev != dev)` `break;` `dst_hold(&rth->u.dst);` `rcu_read_unlock();` `rt = dst_alloc(&ipv4_dst_ops);` `if (rt == NULL) {` `ip_rt_put(rth);` `in_dev_put(in_dev);` `return;` `}` `****` `rt_del(hash, rth);` `if (!rt_intern_hash(hash, rt, &rt))` `ip_rt_put(rt);` `goto do_next;` `}`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			Source: Linux kernel source code file /net/ipv4/route.c As explained above, code within rt_intern_hash performs the function of removing at least some of the expired ones of the records in the linked list. Bedrock contends that that this limitation is literally met both identically and by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
6.	The information storage and retrieval system according to claim 5 further including **means for dynamically determining maximum number** for the record search means to remove in the accessed linked list of records.	**function:** "dynamically determining maximum number for the record search means to remove in the accessed linked list of records" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to dynamically determine a maximum number of records to remove by choosing a search strategy of removing all expired records from a linked list or removing some but not all of the expired records as described in col. 6 line 56 – col. 7 line 15 and/or programmed with software instructions to dynamically determine a maximum number of records to remove by choosing between the pseudocode of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 includes means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of dynamically determining maximum number for the record search means to remove in the accessed linked list of records:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			<pre>if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.rt_next; rt_free(cand); } }</pre> chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: <pre>chain_length++;</pre> Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
7.	A method for storing and retrieving information records using a hashing technique to provide	**external chaining** means "a technique for resolving hash collisions using a linked list(s)"	Bedrock does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
	access to the records and using an **external chaining** technique to store the records with same hash address, at least some of the records **automatically expiring**, the method comprising the steps of:	**automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method for storing and retrieving information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring. The Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to store and retrieve information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, where at least some of the records automatically expire. The Linux IPv4 routing cache use external chaining, a technique for resolving hash collisions using linked lists. In particular, for each unique hash value, the routing cache table, rt_hash_table, contains an entry called a rt_hash_bucket. In turn, a bucket contains an entry named "chain" which is a pointer to the first record of a linked list of routing cache records. `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `  spinlock_t lock;` `} __attribute__((__aligned__(8)));` Source: Linux kernel source code file /net/ipv4/route.c In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c
(a)	accessing a **linked list of records** having same hash address,	**a linked list of records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others'

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
		record"	practice of) a method that includes the step of accessing a linked list of records having same hash address. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to access a linked list of records having same hash address. Specifically, data structure rt_hash_table in module /net/ipv4/route.c is used to access a linked list of records having the same hash address. Additionally, code contained within the function rt_intern_hash in module /net/ipv4/route.c is also used to access a linked list of records having the same hash address. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 practices a method that includes the step of accessing a linked list of records having same hash address. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `  spinlock_t lock;` `} __attribute__((__aligned__(8)));` `struct rtable` `{` `  union` `  {` `  struct dst_entry dst;` `  struct rtable *rt_next;` `  } u;` `  struct in_device *idev;` `  unsigned rt_flags;` `  unsigned rt_type;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			<pre>__u32 rt_dst; /* Path destination */ __u32 rt_src; /* Path source */ int rt_iif; /* Info on neighbour */ __u32 rt_gateway; /* Cache lookup keys */ struct flowi fl; /* Miscellaneous cached information */ __u32 rt_spec_dst; /* RFC1122 specific destination */ struct inet_peer *peer; /* long-living peer info */ };</pre> Source: Linux kernel source code files /net/ipv4/route.c and /include/net/route.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. Because records are hashed to a hash table address before they are added to the chain of records anchored from that address, all records on a chain will have the same hash address. The following code excerpts within route.c performs the step of accessing a linked list of records:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5), tos); if (!rt_intern_hash(hash, rt, &rt)) * or * hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5), tos); return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst); * or * hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5), tos); err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst); * or * hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5), tos); err = rt_intern_hash(hash, rth, rp); static unsigned int rt_hash_code(u32 daddr, u32 saddr, u8 tos) { return (jhash_3words(daddr, saddr, (u32) tos, rt_hash_rnd) & rt_hash_mask); } rt_intern_hash accesses the linked list: rthp = &rt_hash_table[hash].chain; Source: Linux kernel source code file /net/ipv4/route.c
(b)	identifying at least some of the automatically **expired** ones of the records,	**expired** means "obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of identifying at least some of

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			the automatically expired ones of the records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to identify at least some of the automatically expired ones of the records. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. Code contained within or accessed by the function rt_intern_hash in module /net/ipv4/route.c uses rt_score to practice a method that includes the step of identifying at least some of the automatically expired ones of the records. The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: spin_lock_bh(&rt_hash_table[hash].lock); while ((rth = *rthp) != NULL) { if (compare_keys(&rth->fl, &rt->fl)) { **** *rp = rth; return 0; } **** if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score;

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			} } chain_length++; rthp = &rth->u.rt_next; } if (cand) { Source: Linux kernel source code file /net/ipv4/route.c
(c)	**removing at least some of the automatically expired records from the linked list when the linked list is accessed**, and	**removing at least some of the automatically expired records from the linked list** means "adjusting the pointer in the linked list to bypass the previously identified expired records" **when the linked list is accessed** means "both identification and removal of the automatically expired record(s) occurs during the same access of the linked list"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to remove at least some of the automatically expired records from the linked list when the linked list is accessed. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the linked list when the linked list is accessed: if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			* only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.rt_next; rt_free(cand); } } The line of code "*candp = cand->u.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records. Source: Linux kernel source code file /net/ipv4/route.c Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash.
(d)	inserting, retrieving or deleting one of the records from the system following the step of removing.		When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of inserting, retrieving or deleting one of the records from the system following the step of removing. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 is especially adapted to insert, retrieve or delete one of the records from the system following the step of removing. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of inserting one of the records from the system following the step of removing. The following excerpt from the rt_intern_hash function is an example code which practices a method that includes the step of inserting one of the

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			records from the system following the step of removing: `rt->u.rt_next = rt_hash_table[hash].chain;` `****` `rt_hash_table[hash].chain = rt;` Source: Linux kernel source code file /net/ipv4/route.c
	{ordering of the steps}	**ordering of the steps:** The "identifying" step must start before "removal" can begin. However, identification need not be completed before removal can begin. The identification step may overlap with the removal step. The ultimate step of claim 7 must follow or at least partially follow the penultimate step of claim 7.	As shown in the code below, the "identifying" step starts before the "removal" step: `if (cand) {` `/* ip_rt_gc_elasticity used to be average length of chain` `* length, when exceeded gc becomes really aggressive.` `*` `* The second limit is less certain. At the moment it allows` `* only 2 entries per bucket. We will see.` `*/` `if (chain_length > ip_rt_gc_elasticity) {` `*candp = cand->u.rt_next;` `rt_free(cand);` `}` `}` Further, the ultimate step of claim 7 follows the penultimate step of claim 7.
8.	The method according to claim 7 further including the step of **dynamically determining** maximum number of expired ones of the records to remove	**dynamically determining** means "making a decision based on factors internal or external to the information storage and retrieval system"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.11, Google practices (or induces or contributes to others' practice of) a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Computer equipment configured with or utilizing software

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
	when the linked list is accessed.		based on Linux kernel version 2.6.11 is especially adapted to dynamically determine maximum number of expired ones of the records to remove when the linked list is accessed. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.11 practices a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. The following code excerpt from the rt_intern_hash function performs the step of dynamically determining the maximum number of expired ones of the records to remove when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.11
			`chain_length++;` The line of code "if (chain_length > ip_rt_gc_elasticity)" therefore makes a decision based on factors internal or external to the information storage and retrieval system. Source: Linux kernel source code file /net/ipv4/route.c

	Claim Language	**Court's Construction**	**Accused Instrumentalities:** **Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18**
1.	An information storage and retrieval system, the system comprising:		Bedrock Computer Technologies LLC ("Bedrock") does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted for information storage and retrieval.
(a)[1]	**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**,	**a linked list to store and provide access to records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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. Within Linux kernel version 2.6.18, the data structure rt_hash_table in module /net/ipv4/route.c[2] anchors one or more linked list(s) to store and

[1] While the limitations are not lettered in the actual claims of the patent, Bedrock provides them here for ease of reference.

[2] The path names of the cited source code is provided for the defendants' convenience. If any version or customization of Linux kernel version 2.6.18 deviates from the path names that are cited in these charts, such deviations are insignificant because it is the routines, functions, methods, macros, classes, data structures, etc., as embodied on servers and other devices, that infringe.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			provide access to records stored in a memory of the system, at least some of the records automatically expiring. In this way, computer equipment configured with or utilizing software based on 2.6.18 includes 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. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry dst;` `  struct rtable *rt_next;` `  } u;` `  struct in_device *idev;` `  unsigned rt_flags;` `  __u16 rt_type;` `  __u16 rt_multipath_alg;` `  __u32 rt_dst; /* Path destination */` `  __u32 rt_src; /* Path source */` `  int rt_iif;` `  /* Info on neighbour */` `  __u32 rt_gateway;` `  /* Cache lookup keys */` `  struct flowi fl;` `  /* Miscellaneous cached information */`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			`__u32 rt_spec_dst; /* RFC1122 specific destination */` `struct inet_peer *peer; /* long-living peer info */` `};` Source: Linux kernel source code files /net/ipv4/route.c and /include/net/route.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			{ u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system.
(b)	**a record search means utilizing a search key to access the linked list,**	**function:** "utilizing a search key to access the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 31-36 and Boxes 39-41 of FIG. 3 and in col. 5 line 53-col. 6 line 4 and col. 6 lines 14-20, and/or programmed with software instructions as described in the pseudocode of Search Table Procedure (cols. 11 and 12) or	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means utilizing a search key to access the linked list or its equivalent. The following code within route.c performs the function of utilizing a search key to access the linked list:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
		Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	`unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5));` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5));` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `return rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `err = rt_intern_hash(hash, rth, &rtres);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5));` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` `static unsigned int rt_hash_code(u32 daddr, u32 saddr)` `{` `return (jhash_2words(daddr, saddr, rt_hash_rnd) & rt_hash_mask);` `}` rt_intern_hash accesses the linked list and searches for a record by comparing keys: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `#ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED` `if (!(rth->u.dst.flags & DST_BALANCED) &&`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			compare_keys(&rth->fl, &rt->fl)) { #else if (compare_keys(&rth->fl, &rt->fl)) { #endif **** *rp = rth; return 0; } **** chain_length++; rthp = &rth->u.rt_next; } **** spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 && fl1->oif == fl2->oif && fl1->iif == fl2->iif; } Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
(c)	**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**, and	**function:** "identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 33-42 of FIG. 3 and in col. 5 line 53-col. 6 line 34, and/or programmed with software instructions as described in the pseudo-code of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means, 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 or its equivalent. Specifically, code contained within function rt_intern_hash comprises 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 or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `#ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED` `if (!(rth->u.dst.flags & DST_BALANCED) &&` `compare_keys(&rth->fl, &rt->fl)) {` `#else` `if (compare_keys(&rth->fl, &rt->fl)) {` `#endif` `****` `*rp = rth;` `return 0;` `}`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			**** if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score; } } chain_length++; rthp = &rth->u.rt_next; } if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.rt_next; rt_free(cand); } } **** spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 && fl1->oif == fl2->oif && fl1->iif == fl2->iif; } static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30);

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			<pre>if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; }</pre> Source: Linux kernel source code file /net/ipv4/route.c Note that the record(s) identified as expired upon traversal of the linked list is not necessarily the record that rt_intern_hash was called to find. Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash. Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(d)	**means, utilizing the record search means, for accessing the linked list and, at the same time,**	**function:** "utilizing the record search means, accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
	removing at least some of the expired ones of the records in the linked list.	**structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions that provide the insert, retrieve, or delete record capability as described in the flowchart of FIG. 5 and col. 7 line 65 – col. 8 line 32, FIG. 6 and col. 8 lines 33-44, or FIG. 7 and col. 8 lines 45-59, respectively, and/or programmed with software instructions that provide the insert, retrieve or delete record capability as described in the pseudocode of Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9, 10, 11, and 12), or Delete Procedure (cols. 11 and 12), respectively, or the equivalents thereof"	same) a system that is especially adapted to include 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 or its equivalent. The code identified below collectively performs the function of utilizing the record search means, accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list. This function is parsed out below for convenience. The following calls to the hashing function and rt_intern_hash are all utilizations of the record search means: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5));` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5));` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `return rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `err = rt_intern_hash(hash, rth, &rtres);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5));` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` * or * `hash = rt_hash_code(oldflp->fl4_dst,`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			oldflp->fl4_src ^ (oldflp->oif << 5)); err = rt_intern_hash(hash, rth, rp); rt_intern_hash, inserts a record: static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp) { struct rtable *rth, **rthp; unsigned long now; struct rtable *cand, **candp; u32 min_score; int chain_length; int attempts = !in_softirq(); **** rt->u.rt_next = rt_hash_table[hash].chain; #if RT_CACHE_DEBUG >= 2 if (rt->u.rt_next) { struct rtable *trt; printk(KERN_DEBUG "rt_cache @%02x: %u.%u.%u.%u", hash, NIPQUAD(rt->rt_dst)); for (trt = rt->u.rt_next; trt; trt = trt->u.rt_next) printk(" . %u.%u.%u.%u", NIPQUAD(trt->rt_dst)); printk("\n"); } #endif rt_hash_table[hash].chain = rt; spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; rt_intern_hash retrieves a record: static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp) { struct rtable *rth, **rthp; unsigned long now; struct rtable *cand, **candp; u32 min_score; int chain_length; int attempts = !in_softirq();

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			*** /* Put it first */ *rthp = rth->u.rt_next; /* * Since lookup is lockfree, the deletion * must be visible to another weakly ordered CPU before * the insertion at the start of the hash chain. */ rcu_assign_pointer(rth->u.rt_next, rt_hash_table[hash].chain); /* * Since lookup is lockfree, the update writes * must be ordered for consistency on SMP. */ rcu_assign_pointer(rt_hash_table[hash].chain, rth); rth->u.dst.__use++; dst_hold(&rth->u.dst); rth->u.dst.lastuse = now; spin_unlock_bh(rt_hash_lock_addr(hash)); rt_drop(rt); *rp = rth; return 0; } rt_intern_hash is also invoked when deleting a record: unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5)); rthp=&rt_hash_table[hash].chain; rcu_read_lock(); while ((rth = rcu_dereference(*rthp)) != NULL) { struct rtable *rt; if (rth->fl.fl4_dst != daddr || rth->fl.fl4_src != skeys[i] || rth->fl.oif != ikeys[k] || rth->fl.iif != 0) { rthp = &rth->u.rt_next; continue;

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			} if (rth->rt_dst != daddr || rth->rt_src != saddr || rth->u.dst.error || rth->rt_gateway != old_gw || rth->u.dst.dev != dev) break; dst_hold(&rth->u.dst); rcu_read_unlock(); rt = dst_alloc(&ipv4_dst_ops); **** rt_del(hash, rth); if (!rt_intern_hash(hash, rt, &rt)) ip_rt_put(rt); goto do_next; } Source: Linux kernel source code file /net/ipv4/route.c As explained above, code within rt_intern_hash performs the function of accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list. Bedrock contends that that this limitation is literally met both identically and by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
2.	The information storage and retrieval system	**function:** "dynamically determining maximum number for the record search means to remove in the accessed linked list	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
	according to claim 1 further including **means for dynamically determining maximum number** for the record search means to remove in the accessed linked list of records.	of records" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to dynamically determine a maximum number of records to remove by choosing a search strategy of removing all expired records from a linked list or removing some but not all of the expired records as described in col. 6 line 56 – col. 7 line 15 and/or programmed with software instructions to dynamically determine a maximum number of records to remove by choosing between the pseudocode of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 includes means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of dynamically determining maximum number for the record search means to remove in the accessed linked list of records: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			`chain_length++;` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
3.	A method for storing and retrieving information records using a **linked list to store and provide access to the records**, at least some of the records **automatically expiring**, the method comprising the steps of:	**a linked list to store and provide access to the records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	Bedrock does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method for storing and retrieving information records that uses a linked list to store and provide access to the records, where at least some of the records are automatically expiring. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to store and retrieve information records using a linked list to store and provide access to the records, where at least some of the records are automatically expiring. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket  *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable  *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry  dst;` `  struct rtable  *rt_next;` `  } u;` `  struct in_device  *idev;` `  unsigned  rt_flags;` `  __u16  rt_type;` `  __u16  rt_multipath_alg;` `  __u32  rt_dst; /* Path destination  */` `  __u32  rt_src; /* Path source  */` `  int  rt_iif;` `  /* Info on neighbour */` `  __u32  rt_gateway;` `  /* Cache lookup keys */` `  struct flowi  fl;` `  /* Miscellaneous cached information */` `  __u32  rt_spec_dst; /* RFC1122 specific destination` `*/` `  struct inet_peer  *peer; /* long-living peer info */` `};` Source: Linux kernel source code files /net/ipv4/route.c and /include/net/route.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `u32 score = jiffies - rt->u.dst.lastuse;` `score = ~score & ~(3<<30);` `if (rt_valuable(rt))` `score |= (1<<31);` `if (!rt->fl.iif ||` `!(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `score |= (1<<30);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			`return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system.
(a)	accessing the **linked list** of records,	**linked list** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of accessing the linked list of records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to access a linked list of records. Specifically, the data structure rt_hash_table in module /net/ipv4/route.c is used to access the linked list of records. Additionally, code contained within the function rt_intern_hash in module /net/ipv4/route.c is also used to access the linked list of records. The following code excerpts within route.c performs the step of accessing a linked list of records: `unsigned hash = rt_hash_code(daddr,`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			skeys[i] ^ (ikeys[k] << 5)); if (!rt_intern_hash(hash, rt, &rt)) * or * hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5)); return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst); * or * hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5)); return rt_intern_hash(hash, rth, (struct rtable**)&skb->dst); * or * hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5)); err = rt_intern_hash(hash, rth, &rtres); * or * hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5)); err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst); * or * hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5)); err = rt_intern_hash(hash, rth, rp); * or * hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5)); err = rt_intern_hash(hash, rth, rp); rt_intern_hash accesses the linked list: rthp = &rt_hash_table[hash].chain; Source: Linux kernel source code file /net/ipv4/route.c
(b)	identifying at least some of the automatically **expired** ones of the records, and	**expired** means "obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others'

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			practice of) a method that includes the step of identifying at least some of the automatically expired ones of the records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to identify at least some of the automatically expired ones of the records. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			} static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. Code contained within or accessed by the function rt_intern_hash in module /net/ipv4/route.c uses rt_score to practice a method that includes the step of identifying at least some of the automatically expired ones of the records. The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: rthp = &rt_hash_table[hash].chain; spin_lock_bh(rt_hash_lock_addr(hash)); while ((rth = *rthp) != NULL) { #ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED if (!(rth->u.dst.flags & DST_BALANCED) && compare_keys(&rth->fl, &rt->fl)) { #else if (compare_keys(&rth->fl, &rt->fl)) { #endif **** *rp = rth; return 0; }

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			**** if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score; } } chain_length++; rthp = &rth->u.rt_next; } if (cand) { Source: Linux kernel source code file /net/ipv4/route.c
(c)	**removing at least some of the automatically expired records from the linked list when the linked list is accessed**.	**removing at least some of the automatically expired records from the linked list** means "adjusting the pointer in the linked list to bypass the previously identified expired records" **when the linked list is accessed** means "both identification and removal of the automatically expired record(s) occurs during the same access of the linked list"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to remove at least some of the automatically expired records from the linked list when the linked list is accessed. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			linked list when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` The line of code "*candp = cand->u.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records. Source: Linux kernel source code file /net/ipv4/route.c Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash.
	{ordering of the steps}	**ordering of the steps:** The "identifying" step must start before "removal" can begin. However, identification need not be completed before removal can begin. The identification step may overlap with the removal step.	As shown in the code below, the "identifying" step starts before the "removal" step: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			rt_free(cand); } }
4.	The method according to claim 3 further including the step of **dynamically determining** maximum number of expired ones of the records to remove when the linked list is accessed.	**dynamically determining** means "making a decision based on factors internal or external to the information storage and retrieval system"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Specifically, code contained within function rt_intern_hash (in module /net/ipv4/route.c) that dynamically executes based upon comparison with variable ip_rt_gc_elasticity is used to perform the claimed act(s). In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 practices a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to dynamically determine maximum number of expired ones of the records to remove when the linked list is accessed. The following code excerpt from the rt_intern_hash function performs the step of dynamically determining the maximum number of expired ones of the records to remove when the linked list is accessed: `if (cand) {` `/* ip_rt_gc_elasticity used to be average length of chain` `* length, when exceeded gc becomes really aggressive.` `*` `* The second limit is less certain. At the moment it allows` `* only 2 entries per bucket. We will see.` `*/`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			`if (chain_length > ip_rt_gc_elasticity) {` `*candp = cand->u.rt_next;` `rt_free(cand);` `}` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;` The line of code "if (chain_length > ip_rt_gc_elasticity)" therefore makes a decision based on factors internal or external to the information storage and retrieval system. Source: Linux kernel source code file /net/ipv4/route.c
5.	An information storage and retrieval system, the system comprising:		Bedrock Computer Technologies LLC ("Bedrock") does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted for information storage and retrieval.
(a)	**a hashing means to provide access to records stored in a memory of**	**function:** "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	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes,

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
	the system and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring,	records automatically expiring" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to provide a hash table having a pointer to the head of a linked list of externally chained records as described in col. 5 lines 16-26 and/or programmed with software instructions as described in the pseudo-code of Definitions, definition number 4, or the equivalents thereof"	uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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, where at least some of the records are automatically expiring or its equivalent. Specifically, data structure rt_hash_table in module /net/ipv4/route.c implements 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, where at least some of the records automatically are expiring or its equivalent. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry  dst;` `  struct rtable  *rt_next;` `  } u;` `  struct in_device  *idev;` `  unsigned  rt_flags;` `  __u16  rt_type;` `  __u16  rt_multipath_alg;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			`__u32 rt_dst; /* Path destination */` `__u32 rt_src; /* Path source */` `int rt_iif;` `/* Info on neighbour */` `__u32 rt_gateway;` `/* Cache lookup keys */` `struct flowi fl;` `/* Miscellaneous cached information */` `__u32 rt_spec_dst; /* RFC1122 specific destination */` `struct inet_peer *peer; /* long-living peer info */` `};` `****` `ipv4_dst_ops.kmem_cachep = kmem_cache_create("ip_dst_cache", sizeof(struct rtable), 0, SLAB_HWCACHE_ALIGN, NULL, NULL);` `****` `rt_hash_table = (struct rt_hash_bucket *) alloc_large_system_hash("IP route cache", sizeof(struct rt_hash_bucket), rhash_entries, (num_physpages >= 128 * 1024) ? 15 : 17, 0, &rt_hash_log, &rt_hash_mask, 0);` `memset(rt_hash_table, 0, (rt_hash_mask + 1) * sizeof(struct rt_hash_bucket));` Source: Linux kernel source code file /net/ipv4/route.h
(b)	**a record search means utilizing a search key to access a linked list of records having the same hash address**,	**function:** "utilizing a search key to access the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
		lines 22-48, programmed with software instructions as described in Boxes 31-36 and Boxes 39-41 of FIG. 3 and in col. 5 line 53-col. 6 line 4 and col. 6 lines 14-20, and/or programmed with software instructions as described in the pseudocode of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	utilizing a search key to access a linked list of records having the same hash address or its equivalent. The following code within route.c performs the function of utilizing a search key to access the linked list: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5));` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5));` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `return rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `err = rt_intern_hash(hash, rth, &rtres);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5));` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` `static unsigned int rt_hash_code(u32 daddr, u32 saddr)` `{` `return (jhash_2words(daddr, saddr, rt_hash_rnd) & rt_hash_mask);` `}` rt_intern_hash accesses the linked list and searches for a record by

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			comparing keys: `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `#ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED` `if (!(rth->u.dst.flags & DST_BALANCED) &&` `compare_keys(&rth->fl, &rt->fl)) {` `#else` `if (compare_keys(&rth->fl, &rt->fl)) {` `#endif` `****` `*rp = rth;` `return 0;` `}` `****` `chain_length++;` `rthp = &rth->u.rt_next;` `}` `****` `spin_unlock_bh(rt_hash_lock_addr(hash));` `*rp = rt;` `return 0;` `}` `static inline int compare_keys(struct flowi *fl1, struct flowi *fl2)` `{` `return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 &&` `fl1->oif == fl2->oif &&` `fl1->iif == fl2->iif;` `}` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(c)	**the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed**, and	**function:** "identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 33-42 of FIG. 3 and in col. 5 line 53-col. 6 line 34, and/or programmed with software instructions as described in the pseudo-code of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed or its equivalent. Specifically, code contained within function rt_intern_hash comprises 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 or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `#ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED` `if (!(rth->u.dst.flags & DST_BALANCED) &&` `compare_keys(&rth->fl, &rt->fl)) {` `#else` `if (compare_keys(&rth->fl, &rt->fl)) {` `#endif`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			**** *rp = rth; return 0; } **** if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score; } } chain_length++; rthp = &rth->u.rt_next; } if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.rt_next; rt_free(cand); } } **** spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return memcmp(&fl1->nl_u.ip4_u, &fl2->nl_u.ip4_u, sizeof(fl1->nl_u.ip4_u)) == 0 && fl1->oif == fl2->oif && fl1->iif == fl2->iif; }

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c Note that the record(s) identified as expired upon traversal of the linked list is not necessarily the record that rt_intern_hash was called to find. Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash. Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
(d)	**mea[n]s, 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 linked list of records**.	**function:** "utilizing the record search means, 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 linked list of records" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions that provide the insert, retrieve, and delete record capability as described in the flowchart of FIG. 5 and col. 7 line 65 – col. 8 line 32, FIG. 6 and col. 8 lines 33-44, or FIG. 7 and col. 8 lines 45-59, respectively, and/or programmed with software instructions that provide the insert, retrieve and delete record capability as described in the pseudo-code of Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9, 10, 11, and 12), and Delete Procedure (cols. 11 and 12), respectively, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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 linked list of records or its equivalent. The code identified below collectively performs the function of utilizing the record search means, 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 linked list of records. This function is parsed out below for convenience. Specifically, the following calls to the hashing function and rt_intern_hash are all utilizations of the record search means: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5));` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5));` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `return rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `err = rt_intern_hash(hash, rth, &rtres);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5));` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			* or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` rt_intern_hash, inserts a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)` `{` `struct rtable *rth, **rthp;` `unsigned long now;` `struct rtable *cand, **candp;` `u32 min_score;` `int chain_length;` `int attempts = !in_softirq();` `****` `rt->u.rt_next = rt_hash_table[hash].chain;` `#if RT_CACHE_DEBUG >= 2` `if (rt->u.rt_next) {` `struct rtable *trt;` `printk(KERN_DEBUG "rt_cache @%02x: %u.%u.%u.%u", hash, NIPQUAD(rt->rt_dst));` `for (trt = rt->u.rt_next; trt; trt = trt->u.rt_next) printk(" . %u.%u.%u.%u", NIPQUAD(trt->rt_dst));` `printk("\n");` `}` `#endif` `rt_hash_table[hash].chain = rt;` `spin_unlock_bh(rt_hash_lock_addr(hash));` `*rp = rt;` `return 0;` `}` rt_intern_hash retrieves a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			{ struct rtable *rth, **rthp; unsigned long now; struct rtable *cand, **candp; u32 min_score; int chain_length; int attempts = !in_softirq(); *** /* Put it first */ *rthp = rth->u.rt_next; /* * Since lookup is lockfree, the deletion * must be visible to another weakly ordered CPU before * the insertion at the start of the hash chain. */ rcu_assign_pointer(rth->u.rt_next, rt_hash_table[hash].chain); /* * Since lookup is lockfree, the update writes * must be ordered for consistency on SMP. */ rcu_assign_pointer(rt_hash_table[hash].chain, rth); rth->u.dst.__use++; dst_hold(&rth->u.dst); rth->u.dst.lastuse = now; spin_unlock_bh(rt_hash_lock_addr(hash)); rt_drop(rt); *rp = rth; return 0; } rt_intern_hash is also invoked when deleting a record: unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5)); rthp=&rt_hash_table[hash].chain; rcu_read_lock(); while ((rth = rcu_dereference(*rthp)) != NULL) {

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			struct rtable *rt; if (rth->fl.fl4_dst != daddr || rth->fl.fl4_src != skeys[i] || rth->fl.oif != ikeys[k] || rth->fl.iif != 0) { rthp = &rth->u.rt_next; continue; } if (rth->rt_dst != daddr || rth->rt_src != saddr || rth->u.dst.error || rth->rt_gateway != old_gw || rth->u.dst.dev != dev) break; dst_hold(&rth->u.dst); rcu_read_unlock(); rt = dst_alloc(&ipv4_dst_ops); **** rt_del(hash, rth); if (!rt_intern_hash(hash, rt, &rt)) ip_rt_put(rt); goto do_next; } Source: Linux kernel source code file /net/ipv4/route.c As explained above, code within rt_intern_hash performs the function of removing at least some of the expired ones of the records in the linked list. Bedrock contends that that this limitation is literally met both identically and by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			for this term.
6.	The information storage and retrieval system according to claim 5 further including **means for dynamically determining maximum number** for the record search means to remove in the accessed linked list of records.	**function:** "dynamically determining maximum number for the record search means to remove in the accessed linked list of records" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to dynamically determine a maximum number of records to remove by choosing a search strategy of removing all expired records from a linked list or removing some but not all of the expired records as described in col. 6 line 56 – col. 7 line 15 and/or programmed with software instructions to dynamically determine a maximum number of records to remove by choosing between the pseudocode of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 includes means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of dynamically determining maximum number for the record search means to remove in the accessed linked list of records: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
7.	A method for storing and retrieving information records using a hashing technique to provide access to the records and using an **external chaining** technique to store the records with same hash address, at least some of the records **automatically expiring**, the method comprising the steps of:	**external chaining** means "a technique for resolving hash collisions using a linked list(s)" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	Bedrock does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method for storing and retrieving information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring. The Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to store and retrieve information records using a hashing technique to provide access to the records and using an external

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			chaining technique to store the records with same hash address, where at least some of the records automatically expire. The Linux IPv4 routing cache use external chaining, a technique for resolving hash collisions using linked lists. In particular, for each unique hash value, the routing cache table, rt_hash_table, contains an entry called a rt_hash_bucket. In turn, a bucket contains an entry named "chain" which is a pointer to the first record of a linked list of routing cache records. `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `  spinlock_t lock;` `} __attribute__((__aligned__(8)));` Source: Linux kernel source code file /net/ipv4/route.c In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static __inline__ int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c
(a)	accessing a **linked list of records** having same hash address,	**a linked list of records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of accessing a linked list of records having same hash address. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to access a linked list of records having same hash address. Specifically, data structure rt_hash_table in module /net/ipv4/route.c is used to access a linked list of records having the same hash address. Additionally, code contained within the function rt_intern_hash in module /net/ipv4/route.c is also used to access a linked list of records having the same hash address. In this way, computer equipment configured with or

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			utilizing software based on Linux kernel version 2.6.18 practices a method that includes the step of accessing a linked list of records having same hash address. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry  dst;` `  struct rtable  *rt_next;` `  } u;` `  struct in_device  *idev;` `  unsigned  rt_flags;` `  __u16  rt_type;` `  __u16  rt_multipath_alg;` `  __u32  rt_dst; /* Path destination */` `  __u32  rt_src; /* Path source */` `  int  rt_iif;` `  /* Info on neighbour */` `  __u32  rt_gateway;` `  /* Cache lookup keys */` `  struct flowi  fl;` `  /* Miscellaneous cached information */` `  __u32  rt_spec_dst; /* RFC1122 specific destination */` `  struct inet_peer  *peer; /* long-living peer info */`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			}; Source: Linux kernel source code files /net/ipv4/route.c and /include/net/route.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. Because records are hashed to a hash table address before they are added to the chain of records anchored from that address, all records on a chain will have the same hash address. The following code excerpts within route.c performs the step of accessing a linked list of records: `unsigned hash = rt_hash_code(daddr, skeys[i] ^ (ikeys[k] << 5));` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash_code(daddr, saddr ^ (dev->ifindex << 5));` `return rt_intern_hash(hash, rth, (struct rtable**) &skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `return rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl->iif << 5));` `err = rt_intern_hash(hash, rth, &rtres);` * or * `hash = rt_hash_code(daddr, saddr ^ (fl.iif << 5));` `err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			* or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` * or * `hash = rt_hash_code(oldflp->fl4_dst, oldflp->fl4_src ^ (oldflp->oif << 5));` `err = rt_intern_hash(hash, rth, rp);` rt_intern_hash accesses the linked list: `rthp = &rt_hash_table[hash].chain;` Source: Linux kernel source code file /net/ipv4/route.c
(b)	identifying at least some of the automatically **expired** ones of the records,	**expired** means "obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of identifying at least some of the automatically expired ones of the records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to identify at least some of the automatically expired ones of the records. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			Code contained within or accessed by the function rt_intern_hash in module /net/ipv4/route.c uses rt_score to practice a method that includes the step of identifying at least some of the automatically expired ones of the records. The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `#ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED` `if (!(rth->u.dst.flags & DST_BALANCED) &&` `compare_keys(&rth->fl, &rt->fl)) {` `#else` `if (compare_keys(&rth->fl, &rt->fl)) {` `#endif` `****` `*rp = rth;` `return 0;` `}` `****` `if (!atomic_read(&rth->u.dst.__refcnt)) {` `u32 score = rt_score(rth);` `if (score <= min_score) {` `cand = rth;` `candp = rthp;` `min_score = score;` `}` `}` `chain_length++;` `rthp = &rth->u.rt_next;` `}` `if (cand) {`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			Source: Linux kernel source code file /net/ipv4/route.c
(c)	**removing at least some of the automatically expired records from the linked list when the linked list is accessed**, and	**removing at least some of the automatically expired records from the linked list** means "adjusting the pointer in the linked list to bypass the previously identified expired records" **when the linked list is accessed** means "both identification and removal of the automatically expired record(s) occurs during the same access of the linked list"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to remove at least some of the automatically expired records from the linked list when the linked list is accessed. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the linked list when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			The line of code "*candp = cand->u.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records. Source: Linux kernel source code file /net/ipv4/route.c Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash.
(d)	inserting, retrieving or deleting one of the records from the system following the step of removing.		When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of inserting, retrieving or deleting one of the records from the system following the step of removing. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to insert, retrieve or delete one of the records from the system following the step of removing. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of inserting one of the records from the system following the step of removing. The following excerpt from the rt_intern_hash function is an example code which practices a method that includes the step of inserting one of the records from the system following the step of removing: `rt->u.rt_next = rt_hash_table[hash].chain;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			**** `rt_hash_table[hash].chain = rt;` Source: Linux kernel source code file /net/ipv4/route.c
	{ordering of the steps}	**ordering of the steps:** The "identifying" step must start before "removal" can begin. However, identification need not be completed before removal can begin. The identification step may overlap with the removal step. The ultimate step of claim 7 must follow or at least partially follow the penultimate step of claim 7.	As shown in the code below, the "identifying" step starts before the "removal" step: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` Further, the ultimate step of claim 7 follows the penultimate step of claim 7.
8.	The method according to claim 7 further including the step of **dynamically determining** maximum number of expired ones of the records to remove when the linked list is accessed.	**dynamically determining** means "making a decision based on factors internal or external to the information storage and retrieval system"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.18, Google practices (or induces or contributes to others' practice of) a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 is especially adapted to dynamically determine maximum number of expired ones of the records to remove when the linked list is accessed.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.18 practices a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. The following code excerpt from the rt_intern_hash function performs the step of dynamically determining the maximum number of expired ones of the records to remove when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.rt_next;` `  rt_free(cand);` `  }` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.18
			The line of code "if (chain_length > ip_rt_gc_elasticity)" therefore makes a decision based on factors internal or external to the information storage and retrieval system. Source: Linux kernel source code file /net/ipv4/route.c

	Claim Language	**Court's Construction**	**Accused Instrumentalities:** **Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26**
1.	An information storage and retrieval system, the system comprising:		Bedrock Computer Technologies LLC ("Bedrock") does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted for information storage and retrieval.
(a)[1]	**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**,	**a linked list to store and provide access to records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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. Within Linux kernel version 2.6.26, the data structure rt_hash_table in module /net/ipv4/route.c[2] anchors one or more linked list(s) to store and

[1] While the limitations are not lettered in the actual claims of the patent, Bedrock provides them here for ease of reference.

[2] The path names of the cited source code is provided for the defendants' convenience. If any version or customization of Linux kernel version 2.6.26 deviates from the path names that are cited in these charts, such deviations are insignificant because it is the routines, functions, methods, macros, classes, data structures, etc., as embodied on servers and other devices, that infringe.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			provide access to records stored in a memory of the system, at least some of the records automatically expiring. In this way, computer equipment configured with or utilizing software based on 2.6.26 includes 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. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table __read_mostly;` `struct rt_hash_bucket {` `  struct rtable *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry dst;` `  } u;` `  /* Cache lookup keys */` `  struct flowi fl;` `  struct in_device *idev;` `  int rt_genid;` `  unsigned rt_flags;` `  __u16 rt_type;` `  __be32 rt_dst; /* Path destination */` `  __be32 rt_src; /* Path source */` `  int rt_iif;` `  /* Info on neighbour */` `  __be32 rt_gateway;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			`/* Miscellaneous cached information */` `__be32 rt_spec_dst; /* RFC1122 specific destination */` `struct inet_peer *peer; /* long-living peer info */` `};` `struct dst_entry` `{` `struct rcu_head rcu_head;` `struct dst_entry *child;` `struct net_device *dev;` `short error;` `short obsolete;` `int flags;` `#define DST_HOST 1` `#define DST_NOXFRM 2` `#define DST_NOPOLICY 4` `#define DST_NOHASH 8` `unsigned long expires;` `unsigned short header_len; /* more space at head required */` `unsigned short trailer_len; /* space to reserve at tail */` `unsigned int rate_tokens;` `unsigned long rate_last; /* rate limiting for ICMP */` `struct dst_entry *path;` `struct neighbour *neighbour;` `struct hh_cache *hh;` `struct xfrm_state *xfrm;` `int (*input)(struct sk_buff*);` `int (*output)(struct sk_buff*);` `struct dst_ops *ops;` `u32 metrics[RTAX_MAX];` `#ifdef CONFIG_NET_CLS_ROUTE` `__u32 tclassid;` `#endif` `/*` `* __refcnt wants to be on a different cache line from`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			* input/output/ops or performance tanks badly */ atomic_t __refcnt; /* client references */ int __use; unsigned long lastuse; union { struct dst_entry *next; struct rtable *rt_next; struct rt6_info *rt6_next; struct dn_route *dn_next; }; }; Source: Linux kernel source code files /net/ipv4/route.c, /include/net/route.h, and /include/net/dst.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.dst.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.dst.rt_next field contains a null pointer. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local)

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static inline int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system. Another way in which records can expire in the Linux IPv4 routing cache is through the rt_genid identifier, which identifies the generation to which a record belongs. If a record belongs to an older generation, i.e., a generation prior to the current generation, that record is deemed expired.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
(b)	**a record search means utilizing a search key to access the linked list,**	**function:** "utilizing a search key to access the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 31-36 and Boxes 39-41 of FIG. 3 and in col. 5 line 53-col. 6 line 4 and col. 6 lines 14-20, and/or programmed with software instructions as described in the pseudocode of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means utilizing a search key to access the linked list or its equivalent. The following code within route.c performs the function of utilizing a search key to access the linked list: `unsigned hash = rt_hash(daddr, skeys[i], ikeys[k]);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash(daddr, saddr, dev->ifindex);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl->iif);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl.iif);` `err = rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif);` `err = rt_intern_hash(hash, rth, rp);` `static inline unsigned int rt_hash(__be32 daddr, __be32 saddr, int idx)` `{` `return jhash_3words((__force u32)(__be32)(daddr),` `(__force u32)(__be32)(saddr),` `idx, atomic_read(&rt_genid))` `& rt_hash_mask;` `}` rt_intern_hash accesses the linked list and searches for a record by

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			comparing keys: `rthp = &rt_hash_table[hash].chain;` `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `****` `if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) {` `****` `*rp = rth;` `return 0;` `}` `****` `chain_length++;` `rthp = &rth->u.dst.rt_next;` `}` `****` `spin_unlock_bh(rt_hash_lock_addr(hash));` `*rp = rt;` `return 0;` `}` `static inline int compare_keys(struct flowi *fl1, struct flowi *fl2)` `{` `return ((__force u32)((fl1->nl_u.ip4_u.daddr ^ fl2->nl_u.ip4_u.daddr) |` `(fl1->nl_u.ip4_u.saddr ^ fl2->nl_u.ip4_u.saddr)) |` `(fl1->mark ^ fl2->mark) |` `(*(u16 *)&fl1->nl_u.ip4_u.tos ^` `*(u16 *)&fl2->nl_u.ip4_u.tos) |` `(fl1->oif ^ fl2->oif) |` `(fl1->iif ^ fl2->iif)) == 0;` `}` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(c)	**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**, and	**function:** "identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 33-42 of FIG. 3 and in col. 5 line 53-col. 6 line 34, and/or programmed with software instructions as described in the pseudo-code of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means, 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 or its equivalent. Specifically, code contained within function rt_intern_hash comprises 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 or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed: `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `if (rth->rt_genid != atomic_read(&rt_genid)) {` `*rthp = rth->u.dst.rt_next;` `rt_free(rth);` `continue;` `}` `if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) {` `****`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			rp = rth; return 0; } if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score; } } chain_length++; rthp = &rth->u.dst.rt_next; } if (cand) { / ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.dst.rt_next; rt_free(cand); } } **** spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return ((__force u32)((fl1->nl_u.ip4_u.daddr ^ fl2->nl_u.ip4_u.daddr) | (fl1->nl_u.ip4_u.saddr ^ fl2->nl_u.ip4_u.saddr)) | (fl1->mark ^ fl2->mark) | (*(u16 *)&fl1->nl_u.ip4_u.tos ^ *(u16 *)&fl2->nl_u.ip4_u.tos) | (fl1->oif ^ fl2->oif) | (fl1->iif ^ fl2->iif)) == 0; }

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static inline int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c Note that the record(s) identified as expired upon traversal of the linked list is not necessarily the record that rt_intern_hash was called to find. Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash. Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
(d)	**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**.	**function:** "utilizing the record search means, accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions that provide the insert, retrieve, or delete record capability as described in the flowchart of FIG. 5 and col. 7 line 65 – col. 8 line 32, FIG. 6 and col. 8 lines 33-44, or FIG. 7 and col. 8 lines 45-59, respectively, and/or programmed with software instructions that provide the insert, retrieve or delete record capability as described in the pseudocode of Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9, 10, 11, and 12), or Delete Procedure (cols. 11 and 12), respectively, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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 or its equivalent. The code identified below collectively performs the function of utilizing the record search means, accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list. This function is parsed out below for convenience. The following calls to the hashing function and rt_intern_hash are all utilizations of the record search means: `unsigned hash = rt_hash(daddr, skeys[i], ikeys[k]);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash(daddr, saddr, dev->ifindex);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl->iif);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl.iif);` `err = rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif);` `err = rt_intern_hash(hash, rth, rp);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			rt_intern_hash, inserts a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)` `{` `  struct rtable  *rth, **rthp;` `  unsigned long  now;` `  struct rtable *cand, **candp;` `  u32  min_score;` `  int  chain_length;` `  int attempts = !in_softirq();` `****` `  rt->u.dst.rt_next = rt_hash_table[hash].chain;` `#if RT_CACHE_DEBUG >= 2` `  if (rt->u.dst.rt_next) {` `  struct rtable *trt;` `  printk(KERN_DEBUG "rt_cache @%02x: " NIPQUAD_FMT, hash,` `  NIPQUAD(rt->rt_dst));` `  for (trt = rt->u.dst.rt_next; trt; trt = trt->u.dst.rt_next)` `  printk(" . " NIPQUAD_FMT, NIPQUAD(trt->rt_dst));` `  printk("\n");` `  }` `#endif` `  rt_hash_table[hash].chain = rt;` `  spin_unlock_bh(rt_hash_lock_addr(hash));` `  *rp = rt;` `  return 0;` `}` rt_intern_hash retrieves a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)` `{` `  struct rtable  *rth, **rthp;` `  unsigned long  now;` `  struct rtable *cand, **candp;` `  u32  min_score;` `  int  chain_length;` `  int attempts = !in_softirq();` `****` `  /* Put it first */` `  *rthp = rth->u.dst.rt_next;` `  /*`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26

```
                         * Since lookup is lockfree, the deletion
                         * must be visible to another weakly ordered CPU
before
                         * the insertion at the start of the hash chain.
                         */
                        rcu_assign_pointer(rth->u.dst.rt_next,
                                           rt_hash_table[hash].chain);
                        /*
                         * Since lookup is lockfree, the update writes
                         * must be ordered for consistency on SMP.
                         */
                        rcu_assign_pointer(rt_hash_table[hash].chain, rth);

                        dst_use(&rth->u.dst, now);
                        spin_unlock_bh(rt_hash_lock_addr(hash));

                        rt_drop(rt);
                        *rp = rth;
                        return 0;
                }
```

rt_intern_hash is also invoked when deleting a record:

```
                        rthp=&rt_hash_table[hash].chain;

                        rcu_read_lock();
                        while ((rth = rcu_dereference(*rthp)) != NULL) {
                                struct rtable *rt;

                                if (rth->fl.fl4_dst != daddr ||
                                    rth->fl.fl4_src != skeys[i] ||
                                    rth->fl.oif != ikeys[k] ||
                                    rth->fl.iif != 0 ||
                                    rth->rt_genid != atomic_read(&rt_genid) ||
                                    !net_eq(dev_net(rth->u.dst.dev), net)) {
                                        rthp = &rth->u.dst.rt_next;
                                        continue;
                                }

                                if (rth->rt_dst != daddr ||
                                    rth->rt_src != saddr ||
                                    rth->u.dst.error ||
                                    rth->rt_gateway != old_gw ||
                                    rth->u.dst.dev != dev)
                                        break;
```

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			dst_hold(&rth->u.dst); rcu_read_unlock(); rt = dst_alloc(&ipv4_dst_ops); if (rt == NULL) { ip_rt_put(rth); in_dev_put(in_dev); return; } **** rt_del(hash, rth); if (!rt_intern_hash(hash, rt, &rt)) ip_rt_put(rt); goto do_next; } static void rt_del(unsigned hash, struct rtable *rt) { struct rtable **rthp, *aux; rthp = &rt_hash_table[hash].chain; spin_lock_bh(rt_hash_lock_addr(hash)); ip_rt_put(rt); while ((aux = *rthp) != NULL) { if (aux == rt || (aux->rt_genid != atomic_read(&rt_genid))) { *rthp = aux->u.dst.rt_next; rt_free(aux); continue; } rthp = &aux->u.dst.rt_next; } spin_unlock_bh(rt_hash_lock_addr(hash)); } Source: Linux kernel source code file /net/ipv4/route.c As explained above, code within rt_intern_hash performs the function of accessing the linked list and, at the same time, removing at least some of the expired ones of the records in the linked list.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			Bedrock contends that that this limitation is literally met both identically and by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
2.	The information storage and retrieval system according to claim 1 further including **means for dynamically determining maximum number** for the record search means to remove in the accessed linked list of records.	**function:** "dynamically determining maximum number for the record search means to remove in the accessed linked list of records" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to dynamically determine a maximum number of records to remove by choosing a search strategy of removing all expired records from a linked list or removing some but not all of the expired records as described in col. 6 line 56 – col. 7 line 15 and/or programmed with software instructions to dynamically determine a maximum number of records to remove by choosing between the pseudocode of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 includes means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of dynamically determining maximum number for the record search means to remove in the accessed linked list of records: `if (cand) {` `/* ip_rt_gc_elasticity used to be average length of chain` `* length, when exceeded gc becomes really aggressive.` `*`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			* The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.dst.rt_next; rt_free(cand); } } chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: chain_length++; Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
3.	A method for storing and retrieving information records using a **linked list to store and provide access to the records**, at least some of the records **automatically expiring**, the method comprising the	**a linked list to store and provide access to the records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	Bedrock does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method for storing and retrieving information records that

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
	steps of:		uses a linked list to store and provide access to the records, where at least some of the records are automatically expiring. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to store and retrieve information records using a linked list to store and provide access to the records, where at least some of the records are automatically expiring. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket  *rt_hash_table __read_mostly;` `struct rt_hash_bucket {` `  struct rtable  *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry  dst;` `  } u;` `  /* Cache lookup keys */` `  struct flowi  fl;` `  struct in_device  *idev;` `  int  rt_genid;` `  unsigned  rt_flags;` `  __u16  rt_type;` `  __be32  rt_dst; /* Path destination  */` `  __be32  rt_src; /* Path source  */` `  int  rt_iif;` `  /* Info on neighbour */`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			__be32 rt_gateway; /* Miscellaneous cached information */ __be32 rt_spec_dst; /* RFC1122 specific destination */ struct inet_peer *peer; /* long-living peer info */ }; struct dst_entry { struct rcu_head rcu_head; struct dst_entry *child; struct net_device *dev; short error; short obsolete; int flags; #define DST_HOST 1 #define DST_NOXFRM 2 #define DST_NOPOLICY 4 #define DST_NOHASH 8 unsigned long expires; unsigned short header_len; /* more space at head required */ unsigned short trailer_len; /* space to reserve at tail */ unsigned int rate_tokens; unsigned long rate_last; /* rate limiting for ICMP */ struct dst_entry *path; struct neighbour *neighbour; struct hh_cache *hh; struct xfrm_state *xfrm; int (*input)(struct sk_buff*); int (*output)(struct sk_buff*); struct dst_ops *ops; u32 metrics[RTAX_MAX]; #ifdef CONFIG_NET_CLS_ROUTE __u32 tclassid; #endif

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			`/*` ` * __refcnt wants to be on a different cache line from` ` * input/output/ops or performance tanks badly` ` */` `atomic_t __refcnt; /* client references */` `int __use;` `unsigned long lastuse;` `union {` `struct dst_entry *next;` `struct rtable *rt_next;` `struct rt6_info *rt6_next;` `struct dn_route *dn_next;` `};` `};` Source: Linux kernel source code files /net/ipv4/route.c, /include/net/route.h, and /include/net/dst.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.dst.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.dst.rt_next field contains a null pointer. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static inline int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system. Another way in which records can expire in the Linux IPv4 routing cache is through the rt_genid identifier, which identifies the generation to which a

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			record belongs. If a record belongs to an older generation, i.e., a generation prior to the current generation, that record is deemed expired.
(a)	accessing the **linked list** of records,	**linked list** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of accessing the linked list of records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to access a linked list of records. Specifically, the data structure rt_hash_table in module /net/ipv4/route.c is used to access the linked list of records. Additionally, code contained within the function rt_intern_hash in module /net/ipv4/route.c is also used to access the linked list of records. The following code excerpts within route.c performs the step of accessing a linked list of records: `unsigned hash = rt_hash(daddr, skeys[i], ikeys[k]);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash(daddr, saddr, dev->ifindex);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl->iif);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl.iif);` `err = rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif);` `err = rt_intern_hash(hash, rth, rp);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			rt_intern_hash accesses the linked list: `rthp = &rt_hash_table[hash].chain;` Source: Linux kernel source code file /net/ipv4/route.c
(b)	identifying at least some of the automatically **expired** ones of the records, and	**expired** means "obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of identifying at least some of the automatically expired ones of the records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to identify at least some of the automatically expired ones of the records. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static __inline__ int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. Another way in which records can expire in the Linux IPv4 routing cache is through the rt_genid identifier, which identifies the generation to which a record belongs. If a record belongs to an older generation, i.e., a generation prior to the current generation, that record is deemed expired. Code contained within or accessed by the function rt_intern_hash in module /net/ipv4/route.c uses rt_score to practice a method that includes the step of identifying at least some of the automatically expired ones of the records.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `if (rth->rt_genid != atomic_read(&rt_genid)) {` `*rthp = rth->u.dst.rt_next;` `rt_free(rth);` `continue;` `}` `if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) {` `****` `*rp = rth;` `return 0;` `}` `if (!atomic_read(&rth->u.dst.__refcnt)) {` `u32 score = rt_score(rth);` `if (score <= min_score) {` `cand = rth;` `candp = rthp;` `min_score = score;` `}` `}` `chain_length++;` `rthp = &rth->u.dst.rt_next;` `}` `if (cand) {` The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: `if (rth->rt_genid != atomic_read(&rt_genid))`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			Source: Linux kernel source code file /net/ipv4/route.c
(c)	**removing at least some of the automatically expired records from the linked list when the linked list is accessed**.	**removing at least some of the automatically expired records from the linked list** means "adjusting the pointer in the linked list to bypass the previously identified expired records" **when the linked list is accessed** means "both identification and removal of the automatically expired record(s) occurs during the same access of the linked list"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to remove at least some of the automatically expired records from the linked list when the linked list is accessed. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the linked list when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.dst.rt_next;` `  rt_free(cand);` `  }` `}`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			The line of code "*candp = cand->u..dst.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the linked list when the linked list is accessed: `if (rth->rt_genid != atomic_read(&rt_genid)) {` `*rthp = rth->u.dst.rt_next;` The line of code "*candp = cand->u.dst.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records. Source: Linux kernel source code file /net/ipv4/route.c Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash.
	{ordering of the steps}	**ordering of the steps:** The "identifying" step must start before "removal" can begin. However, identification need not be completed before removal can begin. The identification step may overlap with the removal step.	As shown in the code below, the "identifying" step starts before the "removal" step: `}` `if (cand) {` `/* ip_rt_gc_elasticity used to be average length of chain` `* length, when exceeded gc becomes really aggressive.` `*` `* The second limit is less certain. At the moment it allows` `* only 2 entries per bucket. We will see.` `*/` `if (chain_length > ip_rt_gc_elasticity) {` `*candp = cand->u.dst.rt_next;` `rt_free(cand);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			} } and if (rth->rt_genid != atomic_read(&rt_genid)) { *rthp = rth->u.dst.rt_next;
4.	The method according to claim 3 further including the step of **dynamically determining** maximum number of expired ones of the records to remove when the linked list is accessed.	**dynamically determining** means "making a decision based on factors internal or external to the information storage and retrieval system"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Specifically, code contained within function rt_intern_hash (in module /net/ipv4/route.c) that dynamically executes based upon comparison with variable ip_rt_gc_elasticity is used to perform the claimed act(s). In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 practices a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to dynamically determine maximum number of expired ones of the records to remove when the linked list is accessed. The following code excerpt from the rt_intern_hash function performs the step of dynamically determining the maximum number of expired ones of the records to remove when the linked list is accessed:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			} if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.dst.rt_next; rt_free(cand); } } chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: chain_length++; The line of code "if (chain_length > ip_rt_gc_elasticity)" therefore makes a decision based on factors internal or external to the information storage and retrieval system. Source: Linux kernel source code file /net/ipv4/route.c
5.	An information storage and retrieval system, the system comprising:		Bedrock Computer Technologies LLC ("Bedrock") does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			same) a system that is especially adapted for information storage and retrieval.
(a)	**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**,	**function:** "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" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to provide a hash table having a pointer to the head of a linked list of externally chained records as described in col. 5 lines 16-26 and/or programmed with software instructions as described in the pseudo-code of Definitions, definition number 4, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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, where at least some of the records are automatically expiring or its equivalent. Specifically, data structure rt_hash_table in module /net/ipv4/route.c implements 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, where at least some of the records automatically are expiring or its equivalent. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table __read_mostly;` `struct rt_hash_bucket {` `  struct rtable *chain;` `};` `struct rtable` `{` `  union` `  {` `  struct dst_entry  dst;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			} u; /* Cache lookup keys */ struct flowi fl; struct in_device *idev; int rt_genid; unsigned rt_flags; __u16 rt_type; __be32 rt_dst; /* Path destination */ __be32 rt_src; /* Path source */ int rt_iif; /* Info on neighbour */ __be32 rt_gateway; /* Miscellaneous cached information */ __be32 rt_spec_dst; /* RFC1122 specific destination */ struct inet_peer *peer; /* long-living peer info */ }; struct dst_entry { struct rcu_head rcu_head; struct dst_entry *child; struct net_device *dev; short error; short obsolete; int flags; #define DST_HOST 1 #define DST_NOXFRM 2 #define DST_NOPOLICY 4 #define DST_NOHASH 8 unsigned long expires; unsigned short header_len; /* more space at head required */ unsigned short trailer_len; /* space to reserve at tail */ unsigned int rate_tokens; unsigned long rate_last; /* rate limiting for ICMP */ struct dst_entry *path;

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			struct neighbour *neighbour; struct hh_cache *hh; struct xfrm_state *xfrm; int (*input)(struct sk_buff*); int (*output)(struct sk_buff*); struct dst_ops *ops; u32 metrics[RTAX_MAX]; #ifdef CONFIG_NET_CLS_ROUTE __u32 tclassid; #endif /* * __refcnt wants to be on a different cache line from * input/output/ops or performance tanks badly */ atomic_t __refcnt; /* client references */ int __use; unsigned long lastuse; union { struct dst_entry *next; struct rtable *rt_next; struct rt6_info *rt6_next; struct dn_route *dn_next; }; }; ipv4_dst_ops.kmem_cachep = kmem_cache_create("ip_dst_cache", sizeof(struct rtable), 0, SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL); **** rt_hash_table = (struct rt_hash_bucket *) alloc_large_system_hash("IP route cache", sizeof(struct rt_hash_bucket), rhash_entries, (num_physpages >= 128 * 1024) ? 15 : 17, 0, &rt_hash_log, &rt_hash_mask,

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			0); memset(rt_hash_table, 0, (rt_hash_mask + 1) * sizeof(struct rt_hash_bucket)); Source: Linux kernel source code files /net/ipv4/route.c, /include/net/route.h, and /include/net/dst.h
(b)	**a record search means utilizing a search key to access a linked list of records having the same hash address**,	**function:** "utilizing a search key to access the linked list" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 31-36 and Boxes 39-41 of FIG. 3 and in col. 5 line 53-col. 6 line 4 and col. 6 lines 14-20, and/or programmed with software instructions as described in the pseudocode of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include a record search means utilizing a search key to access a linked list of records having the same hash address or its equivalent. The following code within route.c performs the function of utilizing a search key to access the linked list: unsigned hash = rt_hash(daddr, skeys[i], ikeys[k]); if (!rt_intern_hash(hash, rt, &rt)) * or * hash = rt_hash(daddr, saddr, dev->ifindex); return rt_intern_hash(hash, rth, &skb->rtable); * or * hash = rt_hash(daddr, saddr, fl->iif); return rt_intern_hash(hash, rth, &skb->rtable); * or * hash = rt_hash(daddr, saddr, fl.iif); err = rt_intern_hash(hash, rth, &skb->rtable); * or * hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif); err = rt_intern_hash(hash, rth, rp); static inline unsigned int rt_hash(__be32 daddr, __be32 saddr, int idx) { return jhash_3words((__force u32)(__be32)(daddr),

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			(__force u32)(__be32)(saddr), idx, atomic_read(&rt_genid)) & rt_hash_mask; } rt_intern_hash accesses the linked list and searches for a record by comparing keys: rthp = &rt_hash_table[hash].chain; spin_lock_bh(rt_hash_lock_addr(hash)); while ((rth = *rthp) != NULL) { **** if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) { **** *rp = rth; return 0; } **** chain_length++; rthp = &rth->u.dst.rt_next; } **** spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return ((__force u32)((fl1->nl_u.ip4_u.daddr ^ fl2->nl_u.ip4_u.daddr) | (fl1->nl_u.ip4_u.saddr ^ fl2->nl_u.ip4_u.saddr)) | (fl1->mark ^ fl2->mark) | (*(u16 *)&fl1->nl_u.ip4_u.tos ^ *(u16 *)&fl2->nl_u.ip4_u.tos) | (fl1->oif ^ fl2->oif) | (fl1->iif ^ fl2->iif)) == 0; }

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
(c)	**the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed**, and	**function:** "identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4 lines 22-48, programmed with software instructions as described in Boxes 33-42 of FIG. 3 and in col. 5 line 53-col. 6 line 34, and/or programmed with software instructions as described in the pseudo-code of Search Table Procedure (cols. 11 and 12) or Alternate Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed or its equivalent. Specifically, code contained within function rt_intern_hash comprises 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 or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of identifying and removing at least some of the expired ones of the records from the linked list when the linked list is accessed: `spin_lock_bh(rt_hash_lock_addr(hash));` `while ((rth = *rthp) != NULL) {` `if (rth->rt_genid != atomic_read(&rt_genid)) {` `*rthp = rth->u.dst.rt_next;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			rt_free(rth); continue; } if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) { **** *rp = rth; return 0; } if (!atomic_read(&rth->u.dst.__refcnt)) { u32 score = rt_score(rth); if (score <= min_score) { cand = rth; candp = rthp; min_score = score; } } chain_length++; rthp = &rth->u.dst.rt_next; } if (cand) { /* ip_rt_gc_elasticity used to be average length of chain * length, when exceeded gc becomes really aggressive. * * The second limit is less certain. At the moment it allows * only 2 entries per bucket. We will see. */ if (chain_length > ip_rt_gc_elasticity) { *candp = cand->u.dst.rt_next; rt_free(cand); } } **** spin_unlock_bh(rt_hash_lock_addr(hash)); *rp = rt; return 0; } static inline int compare_keys(struct flowi *fl1, struct flowi *fl2) { return ((__force u32)((fl1->nl_u.ip4_u.daddr ^ fl2->nl_u.ip4_u.daddr) | (fl1->nl_u.ip4_u.saddr ^ fl2->nl_u.ip4_u.saddr)) |

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			(fl1->mark ^ fl2->mark) | (*(u16 *)&fl1->nl_u.ip4_u.tos ^ *(u16 *)&fl2->nl_u.ip4_u.tos) | (fl1->oif ^ fl2->oif) | (fl1->iif ^ fl2->iif)) == 0; } static inline u32 rt_score(struct rtable *rt) { u32 score = jiffies - rt->u.dst.lastuse; score = ~score & ~(3<<30); if (rt_valuable(rt)) score |= (1<<31); if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static inline int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; } Source: Linux kernel source code file /net/ipv4/route.c Note that the record(s) identified as expired upon traversal of the linked list is not necessarily the record that rt_intern_hash was called to find. Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash. Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			achieve substantially the same result achieved by the construed structure for this term.
(d)	**mea[n]s, 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 linked list of records**.	**function:** "utilizing the record search means, 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 linked list of records" **structure:** "CPU 10 and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions that provide the insert, retrieve, and delete record capability as described in the flowchart of FIG. 5 and col. 7 line 65 – col. 8 line 32, FIG. 6 and col. 8 lines 33-44, or FIG. 7 and col. 8 lines 45-59, respectively, and/or programmed with software instructions that provide the insert, retrieve and delete record capability as described in the pseudo-code of Insert Procedure (cols. 9 and 10), Retrieve Procedure (cols. 9, 10, 11, and 12), and Delete Procedure (cols. 11 and 12), respectively, or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include 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 linked list of records or its equivalent. The code identified below collectively performs the function of utilizing the record search means, 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 linked list of records. This function is parsed out below for convenience. Specifically, the following calls to the hashing function and rt_intern_hash are all utilizations of the record search means: `unsigned hash = rt_hash(daddr, skeys[i], ikeys[k]);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash(daddr, saddr, dev->ifindex);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl->iif);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl.iif);` `err = rt_intern_hash(hash, rth, &skb->rtable);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			* or * `hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif);` `err = rt_intern_hash(hash, rth, rp);` rt_intern_hash, inserts a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)` `{` `struct rtable *rth, **rthp;` `unsigned long now;` `struct rtable *cand, **candp;` `u32 min_score;` `int chain_length;` `int attempts = !in_softirq();` `****` `rt->u.dst.rt_next = rt_hash_table[hash].chain;` `#if RT_CACHE_DEBUG >= 2` `if (rt->u.dst.rt_next) {` `struct rtable *trt;` `printk(KERN_DEBUG "rt_cache @%02x: " NIPQUAD_FMT, hash, NIPQUAD(rt->rt_dst));` `for (trt = rt->u.dst.rt_next; trt; trt = trt->u.dst.rt_next) printk(" . " NIPQUAD_FMT, NIPQUAD(trt->rt_dst));` `printk("\n");` `}` `#endif` `rt_hash_table[hash].chain = rt;` `spin_unlock_bh(rt_hash_lock_addr(hash));` `*rp = rt;` `return 0;` `}` rt_intern_hash retrieves a record: `static int rt_intern_hash(unsigned hash, struct rtable *rt, struct rtable **rp)` `{` `struct rtable *rth, **rthp;` `unsigned long now;` `struct rtable *cand, **candp;` `u32 min_score;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			int chain_length; int attempts = !in_softirq(); **** /* Put it first */ *rthp = rth->u.dst.rt_next; /* * Since lookup is lockfree, the deletion * must be visible to another weakly ordered CPU before * the insertion at the start of the hash chain. */ rcu_assign_pointer(rth->u.dst.rt_next, rt_hash_table[hash].chain); /* * Since lookup is lockfree, the update writes * must be ordered for consistency on SMP. */ rcu_assign_pointer(rt_hash_table[hash].chain, rth); dst_use(&rth->u.dst, now); spin_unlock_bh(rt_hash_lock_addr(hash)); rt_drop(rt); *rp = rth; return 0; } rt_intern_hash is also invoked when deleting a record: rthp=&rt_hash_table[hash].chain; rcu_read_lock(); while ((rth = rcu_dereference(*rthp)) != NULL) { struct rtable *rt; if (rth->fl.fl4_dst != daddr || rth->fl.fl4_src != skeys[i] || rth->fl.oif != ikeys[k] || rth->fl.iif != 0 || rth->rt_genid != atomic_read(&rt_genid) || !net_eq(dev_net(rth->u.dst.dev), net)) { rthp = &rth->u.dst.rt_next; continue; }

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			if (rth->rt_dst != daddr || rth->rt_src != saddr || rth->u.dst.error || rth->rt_gateway != old_gw || rth->u.dst.dev != dev) break; dst_hold(&rth->u.dst); rcu_read_unlock(); rt = dst_alloc(&ipv4_dst_ops); if (rt == NULL) { ip_rt_put(rth); in_dev_put(in_dev); return; } **** rt_del(hash, rth); if (!rt_intern_hash(hash, rt, &rt)) ip_rt_put(rt); goto do_next; } static void rt_del(unsigned hash, struct rtable *rt) { struct rtable **rthp, *aux; rthp = &rt_hash_table[hash].chain; spin_lock_bh(rt_hash_lock_addr(hash)); ip_rt_put(rt); while ((aux = *rthp) != NULL) { if (aux == rt || (aux->rt_genid != atomic_read(&rt_genid))) { *rthp = aux->u.dst.rt_next; rt_free(aux); continue; } rthp = &aux->u.dst.rt_next; } spin_unlock_bh(rt_hash_lock_addr(hash)); } Source: Linux kernel source code file /net/ipv4/route.c

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			As explained above, code within rt_intern_hash performs the function of removing at least some of the expired ones of the records in the linked list. Bedrock contends that that this limitation is literally met both identically and by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
6.	The information storage and retrieval system according to claim 5 further including **means for dynamically determining maximum number** for the record search means to remove in the accessed linked list of records.	**function:** "dynamically determining maximum number for the record search means to remove in the accessed linked list of records" **structure:** "CPU 10, and RAM 11 of FIG. 1 and col. 3 lines 52-56 and portions of the application software, user access software or operating system software, as described at col. 4, lines 22-48, programmed with software instructions to dynamically determine a maximum number of records to remove by choosing a search strategy of removing all expired records from a linked list or removing some but not all of the expired records as described in col. 6 line 56 – col. 7 line 15 and/or programmed with software instructions to dynamically determine a maximum number of records to remove by choosing between the pseudocode of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), or the equivalents thereof"	When Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google makes, uses, sells, offers to sell or imports (or actively induces or contributes to same) a system that is especially adapted to include means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. Specifically, code contained within function rt_intern_hash, in module /net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 includes means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records or its equivalent. The following code excerpt from the rt_intern_hash function performs the function of dynamically determining maximum number for the record search means to remove in the accessed linked list of records:

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			`if (cand) {` `/* ip_rt_gc_elasticity used to be average length of chain` `* length, when exceeded gc becomes really aggressive.` `*` `* The second limit is less certain. At the moment it allows` `* only 2 entries per bucket. We will see.` `*/` `if (chain_length > ip_rt_gc_elasticity) {` `*candp = cand->u.dst.rt_next;` `rt_free(cand);` `}` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;` Source: Linux kernel source code file /net/ipv4/route.c Bedrock contends that that this limitation is literally met by statutory equivalence per 35 U.S.C. § 112 ¶ 6, i.e., the code in the Accused Instrumentality performs the identical function as construed by the Court, in substantially the same way as the construed structure for this term, to achieve substantially the same result achieved by the construed structure for this term.
7.	A method for storing and retrieving information records using a hashing technique to provide access to the records and using an **external**	**external chaining** means "a technique for resolving hash collisions using a linked list(s)" **automatically expiring** means "becoming obsolete and therefore no longer needed or desired in the storage system	Bedrock does not express a position at this time as to whether the preamble of this claim limits the claim's scope. Nevertheless, Bedrock identifies below aspects of the Accused Instrumentalities that correspond to the claim preamble. When Google uses (or induces or contributes to others' use of) computer

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
	chaining technique to store the records with same hash address, at least some of the records **automatically expiring**, the method comprising the steps of:	because of some condition, event, or period of time"	equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method for storing and retrieving information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring. The Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to store and retrieve information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, where at least some of the records automatically expire. The Linux IPv4 routing cache use external chaining, a technique for resolving hash collisions using linked lists. In particular, for each unique hash value, the routing cache table, rt_hash_table, contains an entry called a rt_hash_bucket. In turn, a bucket contains an entry named "chain" which is a pointer to the first record of a linked list of routing cache records. `static struct rt_hash_bucket *rt_hash_table;` `struct rt_hash_bucket {` `  struct rtable *chain;` `  spinlock_t lock;` `} __attribute__((__aligned__(8)));` Source: Linux kernel source code file /net/ipv4/route.c In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);` `  if (!rt->fl.iif ||` `  !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL)))` `  score |= (1<<30);` `  return score;` `}` `static inline int rt_valuable(struct rtable *rth)` `{` `  return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) ||` `  rth->u.dst.expires;` `}` Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. The records are stored in memory of the information storage system.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			Another way in which records can expire in the Linux IPv4 routing cache is through the rt_genid identifier, which identifies the generation to which a record belongs. If a record belongs to an older generation, i.e., a generation prior to the current generation, that record is deemed expired.Source: Linux kernel source code file /net/ipv4/route.c
(a)	accessing a **linked list of records** having same hash address,	**a linked list of records** means "a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no next record"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of accessing a linked list of records having same hash address. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to access a linked list of records having same hash address. Specifically, data structure rt_hash_table in module /net/ipv4/route.c is used to access a linked list of records having the same hash address. Additionally, code contained within the function rt_intern_hash in module /net/ipv4/route.c is also used to access a linked list of records having the same hash address. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 practices a method that includes the step of accessing a linked list of records having same hash address. The following code excerpts from the files /net/ipv4/route.c and /include/net/route.h show the C language definition for the data structure rt_hash_table and the rtable struct definition used by rt_hash_table: `static struct rt_hash_bucket *rt_hash_table __read_mostly;` `struct rt_hash_bucket {` `struct rtable *chain;`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26

```
};

struct rtable
{
        union
        {
                struct dst_entry      dst;
        } u;

        /* Cache lookup keys */
        struct flowi           fl;

        struct in_device       *idev;

        int                    rt_genid;
        unsigned               rt_flags;
        __u16                  rt_type;

        __be32                 rt_dst; /* Path destination    */
        __be32                 rt_src; /* Path source         */
        int                    rt_iif;

        /* Info on neighbour */
        __be32                 rt_gateway;

        /* Miscellaneous cached information */
        __be32                 rt_spec_dst; /* RFC1122 specific destination
*/
        struct inet_peer       *peer; /* long-living peer info */
};

struct dst_entry
{
        struct rcu_head               rcu_head;
        struct dst_entry       *child;
        struct net_device       *dev;
        short                  error;
        short                  obsolete;
        int                    flags;
#define DST_HOST               1
#define DST_NOXFRM             2
#define DST_NOPOLICY           4
#define DST_NOHASH             8
        unsigned long          expires;

        unsigned short         header len;    /* more space at head required
```

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			/ unsigned short trailer_len; / space to reserve at tail */ unsigned int rate_tokens; unsigned long rate_last; /* rate limiting for ICMP */ struct dst_entry *path; struct neighbour *neighbour; struct hh_cache *hh; struct xfrm_state *xfrm; int (*input)(struct sk_buff*); int (*output)(struct sk_buff*); struct dst_ops *ops; u32 metrics[RTAX_MAX]; #ifdef CONFIG_NET_CLS_ROUTE __u32 tclassid; #endif /* * __refcnt wants to be on a different cache line from * input/output/ops or performance tanks badly */ atomic_t __refcnt; /* client references */ int __use; unsigned long lastuse; union { struct dst_entry *next; struct rtable *rt_next; struct rt6_info *rt6_next; struct dn_route *dn_next; }; }; Source: Linux kernel source code files /net/ipv4/route.c, /include/net/route.h, and /include/net/dst.h In the above code, an entry in the Linux IPv4 routing cache (rt_hash_table) is a list, capable of containing two or more records, in which each record contains a pointer to the next record or information indicating there is no

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			next record. In particular, a rt_hash_table entry contains a field named "chain" which is a pointer to the first record of the list. Records of the list are C structs of the type "rtable". A record contains a field named u.rt_next which is a pointer to the next record in the list. If there is no next record, then the u.rt_next field contains a null pointer. Because records are hashed to a hash table address before they are added to the chain of records anchored from that address, all records on a chain will have the same hash address. The following code excerpts within route.c performs the step of accessing a linked list of records: `unsigned hash = rt_hash(daddr, skeys[i], ikeys[k]);` `if (!rt_intern_hash(hash, rt, &rt))` * or * `hash = rt_hash(daddr, saddr, dev->ifindex);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl->iif);` `return rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(daddr, saddr, fl.iif);` `err = rt_intern_hash(hash, rth, &skb->rtable);` * or * `hash = rt_hash(oldflp->fl4_dst, oldflp->fl4_src, oldflp->oif);` `err = rt_intern_hash(hash, rth, rp);` rt_intern_hash accesses the linked list: `rthp = &rt_hash_table[hash].chain;` Source: Linux kernel source code file /net/ipv4/route.c

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
(b)	identifying at least some of the automatically **expired** ones of the records,	**expired** means "obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of identifying at least some of the automatically expired ones of the records. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to identify at least some of the automatically expired ones of the records. In the Linux IPv4 routing cache, a record of a linked list of the routing cache automatically expires when it becomes obsolete and therefore no longer needed or desired in the storage system because of some condition, event, or period of time. More specifically, Linux IPv4 scores the desirability or need for records in the information storage system based on the following criteria: 1. The age of the routing cache record 2. The type of route (such as multicast, broadcast, and local) 3. If the route has been redirected The function rt_score is the function that scores the desirability or need for records in the information storage system: `static inline u32 rt_score(struct rtable *rt)` `{` `  u32 score = jiffies - rt->u.dst.lastuse;` `  score = ~score & ~(3<<30);` `  if (rt_valuable(rt))` `  score |= (1<<31);`

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			<pre>if (!rt->fl.iif || !(rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST|RTCF_LOCAL))) score |= (1<<30); return score; } static inline int rt_valuable(struct rtable *rth) { return (rth->rt_flags & (RTCF_REDIRECTED | RTCF_NOTIFY)) || rth->u.dst.expires; }</pre> Source: Linux kernel source code file /net/ipv4/route.c At least some of the records—if not all of the records—automatically expire according to the criteria used by rt_score. Another way in which records can expire in the Linux IPv4 routing cache is through the rt_genid identifier, which identifies the generation to which a record belongs. If a record belongs to an older generation, i.e., a generation prior to the current generation, that record is deemed expired.Source: Linux kernel source code file /net/ipv4/route.c Code contained within or accessed by the function rt_intern_hash in module /net/ipv4/route.c uses rt_score to practice a method that includes the step of identifying at least some of the automatically expired ones of the records. The following code excerpt from the rt_intern_hash function performs the step of identifying at least some of the automatically expired ones of the records: <pre>spin_lock_bh(rt_hash_lock_addr(hash)); while ((rth = *rthp) != NULL) {</pre>

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			`if (rth->rt_genid != atomic_read(&rt_genid)) {` `*rthp = rth->u.dst.rt_next;` `rt_free(rth);` `continue;` `}` `if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) {` `****` `*rp = rth;` `return 0;` `}` `if (!atomic_read(&rth->u.dst.__refcnt)) {` `u32 score = rt_score(rth);` `if (score <= min_score) {` `cand = rth;` `candp = rthp;` `min_score = score;` `}` `}` `chain_length++;` `rthp = &rth->u.dst.rt_next;` `}` `if (cand) {` and `if (rth->rt_genid != atomic_read(&rt_genid)) {` Source: Linux kernel source code file /net/ipv4/route.c
(c)	**removing at least some of the automatically expired records from the linked list when the linked list is accessed**, and	**removing at least some of the automatically expired records from the linked list** means "adjusting the pointer in the linked list to bypass the previously identified expired records" **when the linked list is accessed** means "both identification	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. Computer equipment configured with or utilizing software based

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
		and removal of the automatically expired record(s) occurs during the same access of the linked list"	on Linux kernel version 2.6.26 is especially adapted to remove at least some of the automatically expired records from the linked list when the linked list is accessed. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of removing at least some of the automatically expired records from the linked list when the linked list is accessed. The following code excerpt from the rt_intern_hash function is an example of removing at least some of the automatically expired records from the linked list when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.dst.rt_next;` `  rt_free(cand);` `  }` `}` and `if (rth->rt_genid != atomic_read(&rt_genid)) {` `  *rthp = rth->u.dst.rt_next;` The line of code "*candp = cand->u.rt_next;" performs the step of adjusting the pointer in the linked list to bypass the previously identified expired records.

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			Source: Linux kernel source code file /net/ipv4/route.c Both the identification and the removal are performed when the linked list is accessed, as is evidenced by the locking and unlocking of the linked list by rt_intern_hash.
(d)	inserting, retrieving or deleting one of the records from the system following the step of removing.		When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of inserting, retrieving or deleting one of the records from the system following the step of removing. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to insert, retrieve or delete one of the records from the system following the step of removing. Specifically, code contained within the function rt_intern_hash in module /net/ipv4/route.c is used to practice a method that includes the step of inserting one of the records from the system following the step of removing. The following excerpt from the rt_intern_hash function is an example code which practices a method that includes the step of inserting one of the records from the system following the step of removing: `rt->u.rt_next = rt_hash_table[hash].chain;` `****` `rt_hash_table[hash].chain = rt;` Source: Linux kernel source code file /net/ipv4/route.c

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
	{ordering of the steps}	**ordering of the steps:** The "identifying" step must start before "removal" can begin. However, identification need not be completed before removal can begin. The identification step may overlap with the removal step. The ultimate step of claim 7 must follow or at least partially follow the penultimate step of claim 7.	As shown in the code below, the "identifying" step starts before the "removal" step: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.dst.rt_next;` `  rt_free(cand);` `  }` `}` and `if (rth->rt_genid != atomic_read(&rt_genid)) {` `  *rthp = rth->u.dst.rt_next;` Further, the ultimate step of claim 7 follows the penultimate step of claim 7.
8.	The method according to claim 7 further including the step of **dynamically determining** maximum number of expired ones of the records to remove when the linked list is accessed.	**dynamically determining** means "making a decision based on factors internal or external to the information storage and retrieval system"	When Google uses (or induces or contributes to others' use of) computer equipment configured with or utilizing software based on Linux kernel version 2.6.26, Google practices (or induces or contributes to others' practice of) a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 is especially adapted to dynamically determine maximum number of expired ones of the records to remove when the linked list is accessed. Specifically, code contained within function rt_intern_hash, in module

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			/net/ipv4/route.c, dynamically executes based upon comparison with variable ip_rt_gc_elasticity. In this way, computer equipment configured with or utilizing software based on Linux kernel version 2.6.26 practices a method that includes the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed. The following code excerpt from the rt_intern_hash function performs the step of dynamically determining the maximum number of expired ones of the records to remove when the linked list is accessed: `if (cand) {` `  /* ip_rt_gc_elasticity used to be average length of chain` `  * length, when exceeded gc becomes really aggressive.` `  *` `  * The second limit is less certain. At the moment it allows` `  * only 2 entries per bucket. We will see.` `  */` `  if (chain_length > ip_rt_gc_elasticity) {` `  *candp = cand->u.dst.rt_next;` `  rt_free(cand);` `  }` `}` chain_length is a variable that dynamically changes to accurately represent the length of a chain, which is a factor internal to the information storage system: `chain_length++;` The line of code "if (chain_length > ip_rt_gc_elasticity)" therefore makes a decision based on factors internal or external to the information storage and

	Claim Language	Court's Construction	Accused Instrumentalities: Computer equipment configured with or utilizing software based on Linux kernel version 2.6.26
			retrieval system. Source: Linux kernel source code file /net/ipv4/route.c