EXHIBIT B

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

BEDROCK COMPUTER TECHNOLOGIES LLC,	§	
	§	
Plaintiff,	§	
	§	**CASE NO. 6:09-cv-269**
v.	§	
	§	**Jury Trial Demanded**
SOFTLAYER TECHNOLOGIES, INC., et al.	§	
	§	
Defendants.	§	

REBUTTAL REPORT OF DR. MARK JONES

ATTACHMENT 2 - RESPONSE TO LINUX 2.0 ASSERTED ART

1. Validity Analysis

1.1. Organization

I will discuss each claim element individually and will address Dr. Jeffay's and Mr. Williams' arguments on an element-by-element basis. My use of the phrase "Linux 2.0" refers to the asserted prior versions of the Linux kernel, not just Linux 2.0.1.

I incorporate by reference my discussion of the history of Linux in each and every analysis below. For that reason, it should be understood that the specific reasoning that I provide below does not limit my opinion as to why Linux 2.0 does not invalidate the claims of the '120 patent. I also note that, in addition to the documents cited in this discussion, my opinion has been informed by the source code for the versions of Linux as well as KTS0000247 and KTS0000242.

1.2. Independent Claim 1

1.2.1. Term 1(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*

1.2.1.1. Court's Construction

The Court has construed term 1(c):

Claim Term	Court's Construction
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), and equivalents thereof.

1.2.1.2.Analysis

Mr. Williams opines that it is the insertion and subsequent garbage collection on a linked list that meet this step. *See* Williams Report at ¶¶ 25 through 37 (beginning on page 158).[1] Dr. Jeffay, on the other hand, opines that the search for duplicate records is the "access" and that the code for searching for a duplicate and removing expired records meets this limitation. See Jeffay Report at ¶¶ 252. Once again, neither Dr. Jeffay nor Mr. William articulates any theory of statutory equivalence to support their opinions that this claim term is met. In any event, I disagree with both Mr. Williams and Dr. Jeffay. As an initial matter, Linux 2.0 does not perform the identical 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. A hash value is passed as an argument to *rt_cache_add()*, which accesses a linked list to insert a record. After the record is inserted, *rt_cache_add()* releases access of the linked list, and then re-accesses the linked list to perform on-demand garbage collection. That *rt_cache_add()* accesses the linked list twice—one for inserting the record and then again to perform on-demand garbage collection. That there are

[1] Mr. Williams also opines that the invocation of *rt_garbage_collect()* meets this limitation. *See* Williams Report at ¶¶ 39-42 (beginning on page 160). As discussed in my opening report, *rt_garbage_collect()* is an on-demand garbage collection routine and does not come close to performing the function similar to the structure in the Court's construction. Particularly, rt_garbage_collect() has no structure identical to or substantially the combination of the record searching, identification, and removing structures in the Court's construction. Version 2.0.1 of *rt_garbage_collect()* walks the entire hash list looking for expired entries. Expired entries are then deleted when they are found. The garbage collection process stops when the cache size goes below *RT_CACHE_SIZE_MAX* entries, which is hard-wired in the code Linux 2.0.1 to be 256.

two separate accesses is evidenced by the calls to *cli()*, which turns off interrupts thereby preventing other threads from accessing—reading and/or writing—the linked list, and *sti()*, which turn the interrupts back on.[2] Because the identification and removal of expired records does not perform when the linked list is accessed, i.e., using the access from claim 1(b), Linux 2.0 does not perform the identical function. Further, even assuming for the sake of argument that Linux 2.0 performed the identical function, it does not perform the function in an identical or substantially the same way as the structure in the Court's construction. The structures in the Court's construction use the linked list access used for record searching to identify and remove an expired record. *rt_cache_add()*, on the other hand, simply accesses a linked list to insert a record. While it performs garbage collection immediately after this access, it does not piggyback on any accessing that *rt_cache_add()* would do without garbage collection. Put another way, the linked list is accessed by *rt_cache_add()* for no reason other than to perform garbage collection. *See also* Microsoft Computer Dictionary 5ed. (Microsoft Press 2002) (defining the verb access to be "[t]o gain entry to memory in order to read or write data.").

1.3. Dependent Claim 2

1.3.1. Claim language

> *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.*

1.3.1.1. Court's Construction

The Court has construed the following within claim 2:

---

[2] Linux 2.0 also includes the functions *ip_rt_fast_lock()* and *ip_rt_unlock()*. These routines do not actually lock a linked list and instead keep a count of the number of threads which are accessing the entire routing table. Based on the count, these functions will delay certain processes from operating on the routing table, but they do not prevent all threads from accessing the routing table. And again, the count applies to the entire routing table rather than an individual linked list..

Claim Term	Court's Construction
means for dynamically determining maximum number	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 pseudo-code of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), and equivalents thereof.

1.3.1.2.Analysis

As claim 2 depends on claim 1, I incorporate by reference my analysis of claim 1. Dr. Jeffay opines that this limitation is met by Linux 2.0 because Linux 2.0 "contains a conditional statement that determines whether or not to delete an expired record based on a comparison of the record's expiration time with the current time." *See* Jeffay Report at ¶ 263. Mr. Williams, however, points to no dynamic determination for *rt_cache_add()* and instead focused on the *rt_garbage_collect()* routine. *See* Williams at ¶¶ 81-92. I disagree; the structure in Linux 2.0 does not perform the recited function. And once more, both Dr. Jeffay and Mr. Williams ignore the structure in the Court's construction. In any event, the conditional statement that Dr. Jeffay refers to is just a test for expiration. Mr. Williams' conditional statement simply decides whether

to invoke an on-demand garbage collection routine. As such, neither structure in Linux 2.0 performs the identical function for this claim term.

1.4. Independent Claim 5

1.4.1. Term 5(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

1.4.1.1.Court's Construction

The Court has construed term 5(c):

Claim Term	Court's Construction
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	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), and equivalents thereof.

1.4.1.2. Analysis

Mr. Williams opines that it is the insertion and subsequent garbage collection on a linked list that meet this step. *See* Williams Report at ¶ 126 (on page 177). Dr. Jeffay, on the other hand, opines that the search for duplicate records is the "access" and that the code for searching for a duplicate and removing expired records meets this limitation. *See* Jeffay Report at ¶¶ 252. Once again, neither Dr. Jeffay nor Mr. William articulates any theory of statutory equivalence to

support their opinions that this claim term is met. In any event, I disagree with both Mr. Williams and Dr. Jeffay. As an initial matter, Linux 2.0 does not perform the identical 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. A hash value is passed as an argument to *rt_cache_add()*, which accesses a linked list to insert a record. After the record is inserted, *rt_cache_add()* releases access of the linked list, and then re-accesses the linked list to perform on-demand garbage collection. That *rt_cache_add()* accesses the linked list twice—one for inserting the record and then again to perform on-demand garbage collection. That there are two separate accesses is evidenced by the calls to *cli()*, which turns off interrupts thereby preventing other threads from accessing—reading and/or writing—the linked list, and *sti()*, which turn the interrupts back on.[3] Because the identification and removal of expired records does not perform when the linked list is accessed, i.e., using the access from claim 1(b), Linux 2.0 does not perform the identical function. Further, even assuming for the sake of argument that Linux 2.0 performed the identical function, it does not perform the function in an identical or substantially the same way as the structure in the Court's construction. The structures in the Court's construction use the linked list access used for record searching to identify and remove an expired record. *rt_cache_add()*, on the other hand, simply accesses a linked list to insert a record. While it performs garbage collection immediately after this access, it does not piggyback on any accessing that *rt_cache_add()* would do without garbage collection. Put another way, the linked list is accessed by *rt_cache_add()* for no reason other than to perform garbage collection.

[3] Linux 2.0 also includes the functions *ip_rt_fast_lock()* and *ip_rt_unlock()*. These routines do not actually lock a linked list and instead keep a count of the number of threads which are accessing the entire routing table. Based on the count, these functions will delay certain processes from operating on the routing table, but they do not prevent all threads from accessing the routing table. And again, the count applies to the entire routing table rather than an individual linked list..

See also Microsoft Computer Dictionary 5ed. (Microsoft Press 2002) (defining the verb access to be "[t]o gain entry to memory in order to read or write data.").

1.5. Dependent Claim 6

1.5.1. Claim language

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.

1.5.1.1. Court's Construction

The Court has construed the following within claim 6:

Claim Term	Court's Construction
means for dynamically determining maximum number	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 pseudo-code of the Search Table Procedure (cols. 11 and 12) or Alternative Version of Search Table Procedure (cols. 11, 12, 13, and 14), and equivalents thereof.

1.5.1.2. Analysis

As claim 6 depends on claim 5, I incorporate by reference my analysis of claim 5. Dr. Jeffay opines that this limitation is met by Linux 2.0 because Linux 2.0 "contains a conditional

statement that determines whether or not to delete an expired record based on a comparison of the record's expiration time with the current time." *See* Jeffay Report at ¶ 263. Mr. Williams, however, points to no dynamic determination for *rt_cache_add()* and instead focused on the *rt_garbage_collect()* routine. *See* Williams at ¶¶ 81-92 and 152-153. I disagree; the structure in Linux 2.0 does not perform the recited function. And once more, both Dr. Jeffay and Mr. Williams ignore the structure in the Court's construction. In any event, the conditional statement that Dr. Jeffay refers to is just a test for expiration. Mr. Williams' conditional statement simply decides whether to invoke an on-demand garbage collection routine. As such, neither structure in Linux 2.0 performs the identical function for this claim term.