UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte ASWINKUMAR VISHANJI RANA
and COREY ALAN GARROW

Appeal 2009-0158
Application 09/792,267
Technology Center 2600

Decided: December 23, 2008

Before MAHSHID D. SAADAT, ROBERT E. NAPPI,
and KARL D. EASTHOM, *Administrative Patent Judges*.

EASTHOM, *Administrative Patent Judge*.

DECISION ON APPEAL

STATEMENT OF THE CASE

Appellants appeal under 35 U.S.C. § 134 from the Examiner's Final Rejection of claims 1-6 and 9-14. (App. Br. 1).[1] We have jurisdiction under 35 U.S.C. § 6(b).

We affirm.

Appellants claim a method for reordering out of sequence or fragmented data packets or datagrams (PDUs) using a link list. Data packets can become out of order if some of the packets in the sequence are routed differently. Data packets can also become fragmented when routers or switches limit the maximum PDU size. The method extracts header information in each PDU to determine if the data packet is fragmented or out of order. A link list memory maintains certain header data, including *inter alia*, a field holding the head pointer, or address of the first block in a PDU, the length of the last bock of the PDU, and a next field for storing the pointer to the next associated block or PDU. (Spec. 2; 5: 2-6; 10:11-15; Abstract).

Claim 1, illustrative of the invention, follows:

1. A method for reordering out of sequence data packets which are associated with a traffic flow on a data network, the data packets including header information with fields, some data packets including a sequence number that determines packet order, the method comprising:

receiving the data packets from an input;

storing the data packets in a packet memory wherein the packet

[1] The Examiner's Answer (mailed June 20, 2008) ("Ans.") and Appellants' Brief (filed, December 31, 2007) ("App. Br.") and Reply Brief (filed June 5, 2008) ("Reply Br.") detail the respective positions of the parties.

memory is controlled by a link list controller that keeps track of data packets in the packet memory;

extracting desired fields from the data packets including respective sequence numbers; identifying out of sequence data packets;

reordering out of sequence data packets using a link list storing locations of data packets in a traffic flow; and

sending the data packets to an output.

The Examiner relies on the following prior art references to show unpatentability:

Liao	US 6,185,208 B1	Feb. 6, 2001
Chapman	US 6,246,684 B1	Jun. 12, 2001
Bellaton	US 6,473,425 B1	Oct. 29, 2002
Almulhem	US 6,587,431 B1	Jul. 1, 2003

The Examiner rejected:

Claims 1-3, 5, and 6 under 35 U.S.C. § 103(a) based upon the teachings of Chapman and Bellaton;

Claim 4 under 35 U.S.C. § 103(a) based upon the teachings of Chapman, Bellaton, and Almulhem;

Claims 9-11, 13 and 14 under 35 U.S.C. § 103(a) based upon the teachings of Liao and Bellaton; and

Claim 12 under 35 U.S.C. § 103(a) based upon the teachings of Liao, Bellaton, and Almulhem.

ISSUES

Appellants' and the Examiner's arguments and rejections raise the following claim groupings: a) 1-3, 5 and 6; b) 4; c) 9, 13 and 14; d) 10; e) 11; and f) 12. The issues before us are:

1) Did Appellants demonstrate that the Examiner erred in finding that Chapman and Bellaton collectively teach or suggest reordering out of sequence data packets using a link list storing locations of data packets in a traffic flow as set forth in claim 1?[2]

2) Did Appellants demonstrate that the Examiner erred in finding that Chapman, Bellaton and Almulhem collectively teach or suggest "wherein the session id associates the data packets with a particular traffic flow" as set forth in claim 4?

3) Did Appellants demonstrate that the Examiner erred in finding that Liao and Bellaton collectively teach or suggest "extracting desired fields from the fragments respectively including the start offset and the expected offset" and "reassembling the fragments into a complete data packet using a link list storing locations of the fragments making up the data packet" as set

[2] Appellants initially stated that claims 1-3, 5 and 6 stand or fall together (App. Br. 4), but then raised new arguments with respect to claims 2, 3, 5 and 6 (Reply Br. 9-18). Appellants waived these new arguments as not timely raised. Claims 1-3, 5 and 6 remain grouped according to Appellants' initial grouping. *Optivus Tech., Inc. v. Ion Beam Appl'ns S.A.* 469 F.3d 978, 989 (Fed Cir. 2006)("[A]n issue not raised by an appellant in its opening brief . . . is waived.")(citations and quotation marks omitted); *Ex parte Scholl* No. 2007-3653, (BPAI Mar. 13, 2008)((Informative), at 17-18, *available at* http://www.uspto.gov/web/offices/dcom/bpai/its/fd073653.pdf.

forth in claim 9; "using a window data structure to store the start offset and expected offset of respective out of order fragments," as set forth in claim 10; and "comparing the start offset of the respective fragments to an order offset and to the start offset and expected offset of any window entries," as set forth in claim 11?[3]

4) Did Appellants demonstrate that the Examiner erred in finding that Liao, Bellaton, and Almulhem collectively teach or suggest that "reassembling includes [a] time-out mechanism for identifying unreassembled fragments after a prescribed amount of time," as set forth in claim 12?

FINDINGS OF FACT (FF)

1. Chapman discloses completely re-ordering data traffic units or simply reducing the level of miss-ordering to change the units to their original order (col. 2, ll. 64-66; col. 7, ll. 13-18). A data traffic unit includes a part of a packet, a complete packet, a set of packets or an arbitrary set of data bytes (col. 1, ll. 28-31).

2. Re-orderable queue buffers allow changing the order of data packets stored in the respective slots of queue buffers and input buffers temporarily holding the data packets (col. 6, ll. 38-54). Sequence numbers in the data packet headers are used to re-order the packets in the original order. (Col. 7, ll. 42-50).

[3] Appellants do not separately argue dependent claims 13 and 14, grouped with claim 9 (App. Br. 15).

3. One of Chapman's embodiments, employing a single buffer, re-orders traffic units according to sequence numbers, but without taking into account the source of each traffic unit. The first embodiment cannot guarantee complete or correct packet re-ordering because data from different sources can be reordered together (col. 7, ll. 51-67). Another embodiment, employing one buffer per source ID number, re-orders data packets from each source using sequence numbers and the source ID. The second embodiment guarantees "a more definitive re-ordering" than the first. (Chapman, col. 13, ll. 37-54).

4. Bellaton discloses a linked list queue system to control the flow of data packets over a telecommunications network. The linked list queue system, storing pointers to packet data, provides flexibility in size alterations to the queue, and deletions and additions of data, and provides ready comparison of the data in the packets. Packet sequence numbers and lengths are compared to determine if a packet should be deleted or added to the queue. (Col. 5, l. 57 to col. 6, l. 47; Abstract).

5. Bellaton's system works in conjunction with prior art TCP/IP protocols (col. 4, ll. 30-35), including a "sliding window" approach (col. 4, ll. 4-29), that control packet order using, *inter alia*, a "SEQUENCE NO." (to indicate packet position in a message stream), source and destination port numbers, and data offsets "OFF" (required since data options vary), all stored in packet headers (col. 3, l. 65 to col. 4, l. 67; col. 10, ll. 46-50; Fig. 5B). Similarly, for fragmented data, a "FLAG" field in such prior art headers indicates if the data is fragmented. An "IDENT" field uniquely identifies the message packet containing the fragments. A "FRAGMENT OFFSET" field specifies the original position of a fragment in a message,

while "MORE FRAGMENTS" indicates if any more fragments exist. Other fields, such as data "TOTAL LENGTH" also exist in the header. (Bellaton, col. 2, ll. 26-61).

6. Bellaton's duplicate data removal system improves upon such prior art systems. The system extracts and lists header information into linked data queues or registers, including *inter alia* sequence numbers, message length, data offsets, and pointers linking successive data packets. Packets are linked in order by "NEXT" and "PREVIOUS" pointers to packet entries. (Col. 8, l. 43 – col. 9, l. 20; Fig 10). Required data, extracted from incoming packets, is compared in memory queues to existing queue data. Data compared includes sequence data, length data, and source data, to determine if packets are related to the same data flow, where to place the extracted information in the queue, how to order the pointers linking the packets, and whether to drop a data packet. (Bellaton, col. 9, l. 21 to col. 10, l. 24).

7. Liao discloses as background prior art, fragmenting message packets that are too long for a particular wireless network into smaller packets, and then transmitting the packets in order. However, because such fragments do not always arrive in order at a destination, the fragments must be reassembled for transmission and reception using sequence numbers, which designate fragment order within a message, and reference numbers, which designate all the fragments for each particular message. The numbers, *inter alia*, are transmitted in the fragment headers. (Liao, col. 1, l. 22 to col. 2, l. 3).

8. Liao's system improves on the prior art systems by using relatively smaller reference numbers, thereby saving bandwidth and memory. Thus, as in prior art systems, messages in Liao's system are fragmented, and

reassembled in order, using Liao's improved reference number and prior art sequence number techniques. That is, servers or other message transmission/reception units transmit the fragmented messages in correct order using header information for later correct reassembly by a mobile unit. (Col. 2, ll. 25-58; col. 3, ll. 22-34; col. 5, ll. 1-15; *see also generally* col. 5, l. 16 to col. 6, l. 45; *see also* FF 7).

9. For example, Liao's system uses a flag in the last fragment header, and a series number Sn incremented in each successive fragment header in a data packet, each employed to reassemble the fragments in the correct order (col. 5, l. 43 to 46, col. 6, ll. 59-63).

10. Almulhem's system re-orders packets into the proper order using a sorting algorithm/function (Abstract; col. 7, ll. 15-21). The sorting function includes a time window period and a STS I signal signifying order. After the time period expires, the sorting function assumes that a missing data packet expected to be at a "min" memory address in EPQ is lost. On the other hand, a previously unassembled data packet found at the "min" address in EPQ 318 after the timeout is output to EPQ 320 and reassembled. The sorting function then declares the data stream complete and in the proper order and frees up the memory for other functions. (Almulhem, col. 9, ll. 22-28; 59-67; col. 11, l. 46 to col. 12, l. 4).

11. Appellant states that sequence numbers and offsets represent "essentially the same information." (Spec. 11: 9-10). An "expected offset . . . is the start offset of the next fragment in the series." (Spec. 11: 23-27). The offsets are employed with other information to reassemble the fragments. (Spec. 11: 15-27).

12. Appellant's disclosure indicates that windows constitute areas in memory (*see e.g.*, Spec. 10:16-32; Fig. 3E, 3G).

PRINCIPLES OF LAW

"[T]here must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006). "On appeal to the Board, an applicant can overcome a rejection by showing insufficient evidence of *prima facie* obviousness . . . ." *Kahn*, 441 F.3d at 985-986 (*quoting In re Rouffet*, 149 F.3d 1350, 1355 (Fed. Cir. 1998).

"[W]hen a patent 'simply arranges old elements with each performing the same function it had been known to perform' and yields no more than one would expect from such an arrangement, the combination is obvious." *KSR Int'l Co. v. Teleflex, Inc.*, 127 S.Ct. 1727, 1740 (U.S. 2007) (*quoting Sakraida v. Ag. Pro. Inc.*, 96 S.Ct. 1532 (1976)). In ". . . difficult . . . cases . . . the claimed subject matter may involve more than the . . . mere application of a known technique to a piece of prior art ready for the improvement." *Id*. "[O]ne cannot show non-obviousness by attacking references individually where, as here, the rejections are based on combinations of references." *In re Keller*, 642 F.2d 413, 426 (CCPA 1981).

ANALYSIS

Issue 1

Appellants' argument that neither Bellaton nor Chapman teach reordering using a linked list (App. Br. 5-9) amounts to a separate attack on

the references and does not defeat the obviousness rejection under *Keller*, *supra*. As the Examiner generally found, Chapman teaches reordering of packets in a traffic flow by comparing sequence numbers in buffers, while Bellaton teaches using a linked list in the same field of endeavor; i.e., the control of packet data flow in telecommunications systems (FF 1-6, Ans. 4-6, 10-11). Both systems extract sequence numbers from packet headers and compare such numbers, in memory buffers or queues, to control the orderly transmission of packets (FF 1-6). Finally, Bellaton's system orders incoming packets successively using pointers to previous and next packets in a sequence (FF 6). Bellaton's system also drops duplicate packets (FF 6), constituting, or, at a minimum, suggesting, proper re-ordering.

Appellants' quotation and description of Bellaton, alleging a lack of re-ordering, simply demonstrates how Bellaton's system controls packet flow by extracting information from linked packet headers and storing same in linked lists, or queues. (App. Br. 6-7, *quoting* Bellaton, col. 9, ll. 12-20; col. 10, ll.7-32). Using Bellaton's flexible pointer-linked packet header sequence extraction technique (FF 4-6) to replace Chapman's general packet header sequence extraction technique, would have involved no "more than . . . the mere application of a known technique to a piece of prior art ready for the improvement." *KSR*, 127 S. Ct. at 1740. (*Compare* Ans. 11. – Examiner finding that the combination yields predictable results).

Issue 2

Appellants do not dispute that Almulhem teaches "wherein the session id associates the data packets with a particular traffic flow" as set forth in claim 4. Rather, Appellants assert that Chapman teaches away from using such a session id to associate packets with any particular flow, because

Chapman is not concerned with multiple sources of traffic flow. (App. Br. 9-11). Appellants' argument ignores one of Chapman's two different embodiments which re-order packets from multiple sources. (FF 3).

Further, Chapman's embodiment, alleged by Appellants to teach away from identifying sources, does not do so. Rather, as the passage quoted by Appellants demonstrate, Chapman teaches that *not* identifying the source of a multiple source system has "drawbacks" – thus teaching towards identifying the source of a multiple source system (*see* App. Br. 10-11, *quoting* Chapman at col. 7, ll. 51-67; FF 3). Moreover, Chapman specifically teaches "a more definitive re-ordering" by using source ID numbers to account for such multiple sources (FF 3).

Issue 3

Appellants' argument that Liao's system merely fragments a data packet but does not reassemble it, as claim 9 requires (Reply Br. 25), lacks factual support. Liao's system improves on prior art fragmentation processes and necessarily involves correct reassembly of such fragmented data packets (*see* FF 7-9).

Appellants' argument that Liao does not disclose a start offset and an expected offset that helps to determine fragment order, fails to explain how the Examiner's position is in error. (Reply Br. 22-23). As the Examiner explained, Liao's system employs the "current fragment number (start offset) and the total number of fragments (expected offset)." (Ans. 14). The

current number reasonably refers to Liao's sequence/series number (*see* FF 7-9).[4]

The sequence number, by itself, or with the total fragment number, helps Liao's system determine fragment order, and reasonably corresponds to the start offset and expected offset of the total number of fragments in a packet (FF 7-9, *see* Ans. 7, 14). Thus, Liao discloses storing essentially the same offset information as Appellants (*see* FF 11). That is, sequence numbers, like the claimed start and expected offset information, simply identify the relative order of the particular fragment within a message. (FF 7-9, 11). Liao's sequence number necessarily embodies at least three items of data – the previous, current, and next fragment numbers in a message sequence. In any case, Bellaton explicitly discloses using prior art fragment offset data and length data for packet ordering and otherwise. (FF 5).

Moreover, as indicated *supra*, Appellants acknowledge that sequence numbers and offsets provide the same information. (FF 11, *see also* Reply Br. 27 offsets "give information related to the **sequence** of the fragments"). Appellants also state that expected offset merely identifies the next fragment (FF 11) – much like Liao's series number identifies the next fragment in the series. Hence, any distinction between the claimed data and Liao's data either amounts to one based upon nonfunctional descriptive material, or involves the simple substitution of prior art, well-known data elements and their expected functions. In short, Appellants' claimed system functions the

[4] Liao interchanges "series" and "sequence" when referring to the same number.

same as the proposed combination, as the Examiner generally found (Ans. 13-15), even if the claimed system employs different data nomenclatures.

Appellants similarly repeat the limitations of claim 10 and assert that the Examiner erred with respect to Liao's teachings (App. Br 16-17, *quoting* Liao, col. 9, ll. 36-42). Appellants' arguments fail, under *Kahn*, to rebut the Examiner's finding that Liao's system specifically uses stored sequence numbers and/or total fragment number (i.e., start offset and expected offset) of fragments in buffers (i.e., windows), and that the combination generally meets the claim (Ans. 7-9, 16).

For example, Appellants do not explain what constitutes a window structure, or as indicated *supra* with respect to claim 9, start offset or expected offset. Windows, according to Appellants' disclosure, simply constitute areas in memory (FF 12). Bellaton's linked data list queues, storing data contiguously and including sequence numbers and packet lengths, constitute the data and windows recited in claim 10 (FF 6). Similarly, Liao's system memory or buffers, storing *inter alia*, the claimed offsets, i.e., sequences and/or total fragment numbers, also constitute such windows (*see* FF 7-9).

Appellants' assertion that claim 10 requires storing the offsets "only for **<u>respective out of order fragments</u>**" (Reply Br. 27) is not commensurate with the claim scope. The term "only" does not appear in claim 10, and hence Liao's system, storing such data for all the fragments, meets the claim.

With respect to claim 11, Appellants similarly fail to demonstrate error in the Examiner's position. Appellants argue that the various offsets are not compared, (*see* Reply Br. 30-32), but as discussed *supra*, Liao's and Bellaton's sequence numbers, and/or other data, containing essentially the

same information as offsets, are compared. Sequence numbers must be compared to reassemble the fragments (*see* FF 7-9). Further, Bellaton discloses comparing sequence and length numbers in queues or windows, and all manner of prior art fragment offset and other data. (FF 4-6).

Finally, Appellants' arguments that neither Liao or Bellaton disclose the limitation of "reassembling the fragments into a complete data packet using a link list storing locations of the fragments making up the data packet" as set forth in claim 9, using a window as set forth in claim 10, or by comparing as set forth in claim 11, (*see* Reply Br. 23-24, 28, 31-32), amount to separate attacks on the references, which as noted above, under *Keller*, does not demonstrate error in the Examiner's obviousness determination. As noted above, Liao discloses reassembling the fragments, by comparing *inter alia*, sequence numbers in buffers, and Bellaton discloses controlling and comparing data packets in windows using a linked list to store fragment data that provides flexibility. (FF 4-9). Bellaton's pointer system also orders the packets successively with pointers. (*See* FF 6).

Using Bellaton's well-known linked pointer technique to extract Liao's sequence/series numbers from headers and store fragment data (FF 4-9) would have involved no "more than . . . the mere application of a known technique to a piece of prior art ready for the improvement." *KSR*, 127 S. Ct. at 1740. In other words, for claims 9-11, comparing any of the known prior art data types used to control traffic flow and/or order of fragments or packets to related window data amounts to combining predictable prior art techniques. *See KSR supra*.

Issue 4

Appellants argue that Almulhem, which outputs a data packet after a time out, does not meet claim 12, which recites "wherein reassembling includes [sic] time-out mechanism for identifying unreassembled fragments after a prescribed amount of time." However, Almulhem's data packets are not reassembled at EPQ 318 until after the time-out mechanism identifies one as a "min" packet, thereby reasonably meeting the claim as the Examiner generally found. (Ans. 17-18). That is, claim 12, requires "reassembling" to include "identifying unreassembled fragments." Hence, claim 12 reasonably does not preclude reassembling such previously unreassembled packets, identified by the time-out as not yet reassembled, and thereafter reassembled.

Appellants explain that their disclosed system discards any packets still in memory, but not yet reassembled after a time-out mechanism, while Almulhem's system outputs the packet after the time-out mechanism (App. Br. 20-21). That explanation does not distinguish Almulhem's system. If discarding fragments still in memory constitutes identifying fragments, then outputting similar such packet segments still in memory, or designating them for reassembly, also constitutes identifying. Moreover, Almulhem's algorithm similarly assumes that unreassembled packets that should be, but are not, in the "min" location are lost, and also clears the memory thereafter. (FF 10). Designating such unassembled packets as lost, in light of Appellants' disclosed system, which discards unassembled packets, also reasonably constitutes "identifying."

Appellants' assertion that Liao does not disclose "wherein reassembling includes time-out mechanism for identifying unreassembled

fragments after a prescribed amount of time," again, amounts to a separate attack on the references, which, under *Keller*, does not defeat obviousness, as the Examiner found. (Ans. 17-18). As the Examiner also found, one of ordinary skill in the art would have been motivated to implement Almulhem's sorting method in Liao's reassembly method "in order to transmit and receive messages that are larger than the maximum packet size for a network in the proper order." (Ans. 17, *see* FF 7-10).

CONCLUSION

Appellants fail to demonstrate error in the Examiner's findings. Accordingly, we sustain the Examiner's rejections of the claims on appeal.

DECISION

We affirm the Examiner's decision rejecting claims 1-6 and 9-14.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv)(2006).

AFFIRMED

KIS

HAYNES AND BOONE, LLP
IP Section
2323 Victory Avenue
Suite 700
Dallas, TX 75219