UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SANDVINE CORPORATION AND SANDVINE INCORPORATED ULC,
Petitioners,

v.

PACKET INTELLIGENCE, LLC,
Patent Owner.

Case No. IPR2017-00863
U.S. Patent No. 6,665,725

PETITION FOR *INTER PARTES* REVIEW

OF U.S. PATENT NO. 6,665,725

I. INTRODUCTION

Petitioners request *Inter Partes* Review ("IPR") of claims 1-2 of U.S. Patent No. 6,665,725 ("'725 Patent"). EX1033.

II. REQUIREMENTS FOR *INTER PARTES* REVIEW UNDER 37 C.F.R. § 42.104

A. Grounds for Standing Under 37 C.F.R. § 42.104(a)

Petitioners certify that the '725 Patent is available for IPR and that no Petitioner is barred or estopped. Specifically, Petitioners state: (1) they are not the owner of the '725 Patent; (2) have not filed a civil action challenging the validity of any claim of the '725 Patent; (3) this Petition is timely filed less than one year after it was served with a complaint alleging infringement of the '725 Patent; and (4) this Petition is filed more than nine months after the '725 Patent issued.

B. Identification of Challenge Under 37 C.F.R. §42.104(b) and Relief Requested

In view of the prior art, evidence, and discussion of claim limitations, claims 1-2 of the '725 Patent are unpatentable and should be cancelled. 37 C.F.R. §42.104(b)(1). This review is governed by pre-AIA §§102 and 103.

Proposed Statutory Rejections for the '725 Patent
Claims 1-2: Anticipated under §102(b) by WO 97/23076 ("Baker") [EX1038]

1. Level of Ordinary Skill in the Art

A person of ordinary skill in the field of computer networking at the time of the alleged invention, June 30, 1999, ("POSITA") would have had at least a bachelor's degree, or equivalent, in electrical engineering, computer engineering, computer science, or a related field or an equivalent number of years of working experience, and one to two years of experience in networking environments, including at least some experience with network traffic monitors, analyzers, and/or firewalls or other intrusion detection systems. EX1006, *Declaration of Bill Lin, PhD*. ("Lin Decl."), ¶¶20-22.

2. Claim Construction

A claim subject to IPR receives the "broadest reasonable construction in light of the specification of the patent in which it appears." 37 C.F.R. §42.100(b). Unless otherwise noted below, Petitioners propose, for purposes of this proceeding only, that the claim terms of the '725 Patent are presumed to take on their ordinary and customary meaning that the term would have to one of ordinary skill in the art. The claim construction analysis is not, and should not be viewed as a concession by Petitioners as to the proper scope of any claim term in litigation. These assumptions are not a waiver of any argument in any litigation that claim terms in the '725 Patent are indefinite or otherwise invalid or unpatentable.

a. "whereby the data structure is configured for rapid searches using the index set" (claim 1)

Petitioners submit that the term "rapid" is not entitled to patentable weight because it is a statement of a desired result, rather than an apparatus or specific structure to accomplish the desired result. *Hewlett-Packard Co. v. Bausch & Lomb Inc.*, 909 F.2d 1464, 1468 (Fed. Cir. 1990)("[A]pparatus claims cover what a device is, not what a device does."); *In re Schreiber*, 128 F.3d 1473, 1478–79 (Fed. Cir. 1997)("choosing to define an element functionally, i.e., by what it does, carries with it a risk," as functional language is not given patentable weight if the prior art structure can inherently perform the function); *Euramax International, Inc. v. Invisaflow*, LLC, IPR2016–00423, Paper No. 9 at 8-9 (PTAB June 1, 2016)(instituting IPR proceeding, language describing intended use of apparatus not entitled to patentable weight). Affording no patentable weight to language describing an intended use or desired result of an apparatus has long been the rule. *In re Gardiner*, 171 F.2d 313, 315–16 (C.C.P.A. 1948)("It is trite to state that the patentability of apparatus claims must be shown in the structure claimed and not merely upon a use, function, or result thereof."). Accordingly, these limitations are purely functional and should be afforded no patentable weight.

III. OVERVIEW OF THE '725 PATENT

A. Description of the '725 Patent

The '725 Patent was filed on June 30, 2000 and claims priority to a U.S. Provisional App. No. 60/141,903 (EX1005) filed on June 30, 1999. On its face, the '725 Patent is directed to a method of performing protocol specific operations on a packet. EX1033, Abstract. The method includes performing various functions such as receiving the packet and a set of protocol descriptions for protocols that may be used in the packet and performing protocol specific operations on the packet based on the protocol descriptions. *Id*. The protocol description includes child protocols, information related to the location of the child protocol, and any protocol specific operations to be performed on the packet for the particular protocol at a particular layer level. *Id*. The protocol specific operations may also include state processing operations. *Id*.

B. Prosecution History of the '725 Patent

The '725 Patent was filed on June 30, 2000 with 18 claims. *See* EX1039, June 30, 2000 As-Filed Application. The Examiner rejected claims 1 and 16 under §112 as indefinite and claims 1-3, 13-14, and 17-18 as anticipated by Bruell. *Id.* at 06/04/03 Rejection. The Examiner indicated that claims 4-11 and 15-16 contained allowable subject matter, noting that the prior art did not disclose:

- Storing a database in a memory, the database generated from the set of protocol descriptions and including a data structure containing

information on the possible protocols and organized for locating the child protocol related information for any protocol, the data structure contents indexed by a set of one or more indices, the database entry indexed by a particular set of index values including an indication of validity, wherein the child protocol related information includes a child recognition pattern, wherein step (c) of performing the protocol specific operations includes, at any particular protocol layer level starting from the base level, searching the packet at to the particular protocol for the child field, the searching including indexing the data structure until a valid entry is found, and whereby the data structure is configured for rapid searches using the index set.

- Looking up a flow-entry database comprising one or more flow-entries, at least one flow-entry for each previously encountered conversational flow, the looking up using at least some of the selected packet portions and determining if the packet matches an existing flow-entry; if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and if the packet is of a new flow, storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry, wherein the parsing and extraction operations depend on the contents of one or more packet headers.

Id.

In response, Applicants amended the claims based on the indicated allowable subject matter. *Id.* at 06/13/03 Response. Applicants also argued that Bruell does not disclose: protocol descriptions that "describe the protocol

description operations to be performed," "the extraction being to form a function of the selected portions for identifying the packet as belonging to conversational flow," or "processing that is a function of the state of the flow of the packet." *Id.* at pp.10-13; *also id.* at 06/27/03 Supplemental Response (amending claim 18). The amended claims were allowed. *Id*. at 07/01/03 Notice of Allowance.

IV. THERE IS A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE

The following prior art references disclose each limitation of the Challenged Claims. As such, the Challenged Claims are unpatentable. Included below are exemplary citations to the prior art references.

A. Baker Anticipates Claims 1-2 under §102(b)

Baker was filed December 13, 1996, published June 26, 1997, is prior art under §102(b) and, as described below, anticipates Claims 1-2 of the '725 Patent.

Claim 1. A method of performing protocol specific operations on a packet passing through a connection point on a computer network, the method comprising:

Baker discloses a network interface system programmed with analysis control logic for retrieving network frames (i.e., packets) passing through a connection point on a computer network and, for example, performing parsing and extraction operations on the frame (i.e., protocol specific operations) in accordance with protocol description files that describe network protocols and include rules for analyzing them:

"Referring now to FIG. 1, a network interface system in accordance with one form of the present invention, generally referred to as 10, may be implemented in a network device including input devices 12, data storage devices 14, analysis control logic 16 for facilitating the input, storage, retrieval, and analysis of network frames, and output devices 18 for forwarding frames or displaying or printing the results of analyses. A data storage device 14 may include a data file 20 of network frames having n protocol data records, wherein each data record contains data stored in a plurality of predefined fields. Protocol description files 22 also may be stored in the data storage device 14. The protocol description files 22 may include a protocol control record and n field sub-records, which together may describe a subset of a network protocol and include rules for analyzing that protocol.

The network device control logic 16 is capable of retrieving a subset of network frames from the input devices 12 or data files 20 which satisfy one or more criteria based upon extracted field values and filtering criteria contained in one or more of the protocol description files 22. The network device control logic 16 also includes logic for determining frame and protocol header lengths, gathering statistics, verification and error checking, determining routes, varying values, and formatting output."

EX1038, *Baker* at 10:10-35; *also id*. at 3:32-4:9 ("The present invention is directed to improved systems and methods for parsing, filtering, generating and analyzing data (or frames of data) transmitted over a data communications network."), 3:32-35 ("The system of the present invention also preferably includes logic for extracting field values from particular network frames, performing validation and

error checking, and making parsing decisions based upon field values and information in the programmably configurable protocol descriptions."), 1:8-2:10, 2:28-3:8, 3:26-4:33, 26:26-27:35, 30:10-34, 37:17-28, Figs. 11-14, 16; *see also* Elements 10(b)(iii), 10(c).

[1(a)] receiving the packet;

Baker discloses a network interface system, such as a network analyzer, bridge, or router, capable of receiving frames from a network:

> *"[T]he system of the present invention may be incorporated in a network device, such as a network analyzer, bridge, router, or traffic generator, including a CPU and a plurality of input devices, storage devices, and output devices, wherein frames of network data may be received from an associated network, stored in the storage devices, and processed by the CPU based upon one or more programmably configurable protocol descriptions also stored in the storage devices."*

Id. at 4:27-35; *also* 1:8-2:10, 3:32-4:9, 10:10-35, 11:6-8, 37:17-23, Figs. 1, 11.

[1(b)] receiving a set of protocol descriptions for a plurality of protocols that conform to a layered model,

According to the '725 Patent, a protocol description is "source-code information" in the form of, for example, a file. EX1033, *'725 Patent* at 14:63-65; *also id.* at 44:14-22, 42:46-43:9, columns 56-93 (exemplary protocol description language files). Similarly, Baker discloses a set of protocol-specific protocol description files (PDFs) (i.e., protocol descriptions), each corresponding to one of a

plurality of protocols (e.g., Ethernet, IP, TCP) that conform to a layered model, such as the well-known ISO/OSI model.

> *"The protocol description files 22 may include a protocol control record and n field sub-records, which together may describe a subset of a network protocol and include rules for analyzing that protocol."*

EX1038 at 10:22-25, 19:7-10 ("[E]ach of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF).").

> *"The Ethernet protocol definition described above specifies only one possible upper level protocol, the Generic Protocol (GP) which is indicated by placing a hexadecimal 0×8888 value in the protocol type field. The Generic Protocol (GP) specification shown below in Table 13 has been fabricated to provide examples of different types of field functionalities found in actual network protocols."*

EX1038 at 21:32-22:5; *also id.* at 34:16-21 (mentions supporting IP, TCP, UDP, and IPX protocols), 12:22-24 ("It will be appreciated by those skilled in the art that any possible organization of fields for any possible protocol specification is contemplated for the network interface system 10 of the present invention."), 24:11-12 (describing "ParseLvl" variable that is the "Zero based protocol level in ISO reference model of protocol being parsed (current protocol)"), 36:28-36, 38:4-16, Tables 12, 13, Figs. 4, 4A-4D, 5, 5A-5E; EX1033 at 6:32-54; EX1006, ¶179.

Baker discloses that network interface system stores the programmably configurable PDFs (EX1038 at 10:21-22, Fig. 1) and is programmed to receive PDFs and run an initialization/compilation process (i.e., receiving a set of protocol descriptions) as follows:

> *"[T]he initialization of the system includes a determination of the presence of PDF files and the extraction of the protocol and associated control record information from all of the PDF files found. The number of PDF files is determined, and a ProtocolList is constructed consisting of a sorted vector of protocol records at least the size of the number of PDF files found. The name of each protocol record found is inserted in the ProtocolList. The PDF files are then read to memory in the sequence described above for the PDF file writing. The lookup records that indicate a next protocol are associated with the appropriate entries in the ProtocolList."*

EX1038 at 20:35-21:11; *also* 4:27-35, 19:13-20:34; EX1006, ¶¶143-156, 219. There is no disclosure in the '725 Patent of "receiving a set of protocol descriptions" aside from inputting a set of PDL files to a compilation process of a network interface system to generate/build/load/fill internal data structures, which, as discussed above and with respect to claim 2, is disclosed by Baker. EX1033 at 41:15-52, 43:10-58, 9:22-27, 9:42-44, 9:53-54, 22:7-10, 14:63-15:17, Fig. 4; EX1006, ¶¶133, 218-219. *See also* Baker applied to Claim 2.

[1(b)(i)(1)] a protocol description for a particular protocol at a particular layer level including: (i) if there is at least one child protocol of the protocol at the particular layer level, the-one or more child protocols of the particular protocol at the particular layer level,

Each PDF file (i.e., protocol description for a particular protocol at a particular layer level) includes, if there is at least one child protocol, the child protocol(s). Namely, each PDF includes "a protocol control record, and a plurality (n) of field data records." EX1038 at 12:25-28. When initialized in memory, a field sub-record may include a "ptr2np" attribute that includes a "pointer to lookup structure/class . . . next protocol definition to use (0 = none)" and the "next protocol lookup records" are described with reference to Table 4 as including a "Protocol" attribute that is a "pointer to protocol description structure." *Id.* at Tables 2, 4, 14:23-25. Each PDF file includes field sub-record information, including a field record description of fields that may require a next protocol lookup:

> *"In the presently preferred embodiment, the following sequence of data is written to a PDF: ...*
>
> *For the Field Record ...*
>
> *the length of the field in bits,*
>
> *the byte offset from the start of the protocol header, ...*
>
> *if the pointer to the lookup structure/class is not NULL,*
>
> *a call is made to write the lookup type, the number of lookups, and the lookup values"*

Id. at 19:11-20:31; *also* 147:28-148:13 (initialization routine for reading field sub-record elements from PDF, including pointer to lookup structure/class), 177:4-26 (initialization routine for reading multi-protocol lookup structure elements from PDF), EX1006, ¶¶150-156.

Baker provides two exemplary protocol descriptions – for Ethernet and for a fabricated Generic Protocol ("GP"), which is described as being the "only one possible upper level protocol" for Ethernet. EX1038 at 21:32-22:5, Tables 12, 13, Figs. 4-4D, 5-5D. As discussed above, and with respect to Element 1(b), the PDFs for these exemplary protocols would include the information depicted in Figs. 4-4D and 5-5E.

Ethernet[1]: The Ethernet PDF (i.e., protocol description) for the Ethernet protocol (i.e., a particular protocol at a particular layer level) includes one child

[1] Although Baker's protocol descriptions for Ethernet and GP are used as examples throughout this Petition, Baker also discloses supporting additional protocols. EX1038 at 21:32-22:5, 34:16-21, 12:21-34, 25:8-12, 36:28-36; EX1006, ¶180. As described in Baker, each supported protocol would include its own PDF (i.e., protocol description) that includes information for parsing a packet in accordance with the known standards for the protocol. EX1038 at 12:21-33, 4:7-21.

protocol, "GP" (i.e., including … the-one or more child protocols of the particular protocol at the particular layer level):

Ethernet Type Lookup Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	5	0x0000	0x8887	ALL	"Unknown"
Figure 5	5	0x8888	0x8888	ALL	"GP"
[None]	5	0x8889	0xFFFF	ALL	"Unknown"

Figure 4d

Id. at Fig. 4D; *see also* 8:21-30, 21:32-22:5, Table 12, Figs. 4, 4A (Type field references Fig. 4D lookup structure). *See also* Elements 1(b)(i)(2), 1(b)(ii)-(iii).

GP: The GP PDF (i.e., protocol description) for the GP protocol (i.e., a particular protocol at a particular layer level) includes four child protocols, "GP1," "GP2," "GP3," and "GP4" (i.e., the-one or more child protocols of the particular protocol at the particular layer level):

Frame Type Next Protocol Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	4	0x00	0x00	ALL	"Illegal Protocol"
GP1	4	0x01	0x01	ALL	"GP1"
GP2	5	0x02	0x02	ALL	"GP2"
[None]	4	0x03	0xFF	ALL	"Illegal Protocol"

Figure 5c

Source Socket Next Protocol Structure					
Protocol	**Next Index**	**Minimum**	**Maximum**	**Mask**	**Translation**
[None]	8	0x0000	0x0142	ALL	"Unknown Protocol"
GP3	8	0x0143	0x018F	ODD	"GP3"
GP4	8	0x0143	0x018F	EVEN	"GP4"
[None]	8	0x0190	0xFFFF	ALL	"Illegal Protocol"

Figure 5d

Destination Socket Next Protocol Structure					
Protocol	**Next Index**	**Minimum**	**Maximum**	**Mask**	**Translation**
[None]	9	0x0000	0x0142	ALL	"Unknown Protocol"
GP3	9	0x0143	0x018F	ODD	"GP3"
GP4	9	0x0143	0x018F	EVEN	"GP4"
[None]	9	0x0190	0xFFFF	ALL	"Illegal Protocol"

Figure 5e

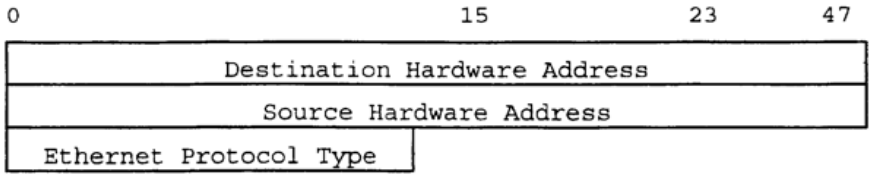
Id. at Figs. 5C-5E; *see also* 8:31-9:6, Table 13 (Frame Type field indicates "upper level protocol identifier," Src Socket and Dst Socket fields indicate "Socket of Upper-layer protocol"), Figs. 5, 5A (Frame field references Fig. 5C lookup structure, Source Socket field references Fig. 5D lookup structure, Destination Socket field references Fig. 5E lookup structure). *See also* Elements 1(b)(i)(2), 1(b)(ii)-(iii).

[1(b)(i)(2)] the packet including for any particular child protocol of the particular protocol at the particular layer level information at one or more locations in the packet related to the particular child protocol,

Ethernet: Baker discloses an "Ethernet Protocol specification" that "is a simplification of an actual Ethernet protocol header":

TABLE 12

ETHERNET v2.0 PROTOCOL SPECIFICATION



Destination Hardware Address	-	destination hardware station address (48 bits)
Source Hardware Address	-	source hardware station address (48 bits)
Ethernet Protocol Type	-	upper layer protocol designator (16 bits) 0x8888=GP

Id. at Table 12; *see also* 21:14-16. According to the disclosed specification, the "Ethernet Protocol Type" field of the Ethernet frame header includes an "upper layer protocol designator 0x8888=GP" (i.e., information at one or more locations in the packet related to the particular child protocol). *Id*.

Additionally, each Ethernet frame includes an Ethernet header that is 14-bytes (112-bits) long followed by the header of the next layer protocol (GP Header) (i.e., information at a location in the packet related to the particular child protocol). *Id.* at 25:29-32 ("Frame 1 shown below has a hardware length of eighty-two 8-bit bytes and consists of a fourteen byte Ethernet header, a twenty byte GP header with no option bytes, and forty-eight bytes of application data.").

Frame (1)

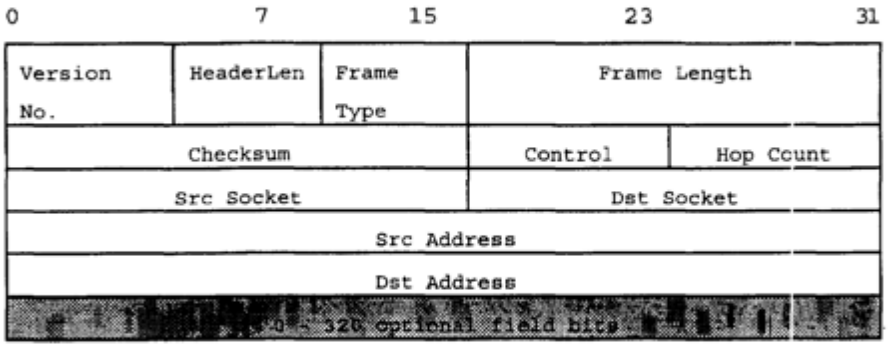
(1)	08	00	00	00	00	03	08	00	00	00	00	04	88			
	88												Ethernet Header(14)			
(2)	35	00	00	44	B1	5F	00	01	08	00	01	47	01	02	03	04
	05	06	07	08									GP Header(20)			
(3)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00						Data (24)		
(4)	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00						Data (24)		

Id. at Frame (1); *also* 26:10-14, Frame (2), 20:32-34. *See also* Elements 1(b)(i)(1), 1(b)(ii)-(iii).

GP: The Baker General Protocol specification is provided for reference below:

TABLE 13

GENERIC PROTOCOL (GP) SPECIFICATION



Version Number (4 bits) - software version number

HeaderLen (4 bits) - length of GP header in 32 bit words.
0- 4 = illegal
5 = No Optional fields,
6-15 = 32-320 bits of options

Frame Type (8 bits) - upper level protocol identifier
0 = Unknown
1 = GP1
2 = GP2
3-255 = Unknown

Frame Length (16 bits) - frame length in bytes including header

Checksum (16 bits) - Checksum of header including options

Control (8 bits) - reserved (must be zero)

Hop Count (8 bits) - Count of number of networks traversed

Src Socket (16 bits) - Socket of Upper-layer protocol sender

Dst Socket (16 bits) - Socket of Upper-layer protocol receiver

Src Address (32 bits) - Sender protocol address

Dst Address (32 bits) - Receiver protocol address

Id. at Table 13; *see also* 21:32-22:5. According to the specification, the Frame Type, Src Socket, and Dst Socket fields provide information regarding the upper-layer/level protocol(s) used in a packet (i.e., information at a location in the packet related to the particular child protocol). *Id*.

Additionally, an Ethernet frame (as disclosed in Baker) may include a forty-eight-byte application data field following the 20-byte GP layer header (i.e., information at a location in the packet related to the particular child protocol). *Id.* at 25:29-32 ("Frame 1 shown below has a hardware length of eighty-two 8-bit bytes and consists of a fourteen byte Ethernet header, a twenty byte GP header with no option bytes, and forty-eight bytes of application data."); *see also* Frame (1). *See also* Elements 1(b)(i)(1), 1(b)(ii)-(iii).

[1(b)(ii)] (ii) the one or more locations in the packet where information is stored related to any child protocol of the particular protocol, and

Each PDF protocol control record may include a "NumBits" attribute that is "the total bit length of the protocol header." EX1038 at Table 1 ("numBits" attribute), 19:17; 56:30-57:11 (initialization routine for reading protocol control record elements from PDF, including "num_bits" attribute), EX1006, ¶¶146-148. The '725 patent similarly discloses a "HEADER" attribute of a PDL file "used to describe the length of the protocol header." EX1033 at 48:41-50. During the parsing process, the value of the numBits attribute may be used to determine where the next layer protocol header/data begins. *See* Element 1(c).

Each PDF additionally includes field sub-records for each field of the protocol header. EX1038 at 12:25-28, Tables 1, 2. Each field sub-record includes an "fdwoff" attribute (also called Byte/Bit Offset) that is the byte offset of the field from the start of the protocol header. EX1038 at Table 2 ("fdwoff" attribute), 20:5, Figs. 4A, 5A; 147:28-148:13 (initialization routine for reading field sub-record elements from PDF, including "fdwoff" attribute), EX1006, ¶¶149-150. This attribute is used during the parsing process to extract the value of the associated field. EX1038 at 27:17-35, Fig. 13 (step 210), Fig. 16 (step 502). *See also* Element 1(c).

Ethernet: The Ethernet PDF NumBits attribute of the protocol control record is set to 112, indicating the total bit length of the Ethernet header:

Ethernet Control Record					
Protocol Name	NumBits	NumFields	CurField	Fields	Options
Ethernet MAC Header	112	5	0	Figure 4a	(None)

Figure 4

EX1038 at Fig. 4.

The Ethernet PDF "Type" field sub-record includes a "Bit Offset" (i.e., fdwoff attribute), indicating that the Type field is 96-bits offset from the start of the Ethernet header:

Index	Field Name	Bit Offset	Bit Length	Left Shift	Right Shift	Check sum	Frame Length	Header Length	Statistics	Lookup Structure	Filter Structure	Format
0	Dst Vendor Address	0	24	0	16	0	0	0	[None]	Fig 4b	Fig 10.Idx 0	nex
1	Dst Station Address	24	24	0	16	0	0	0	[None]	[None]	Fig 10.Idx 1	nex
2	Src Vendor Address	48	24	0	16	0	0	0	[None]	Fig 4c	[None]	nex
3	Src Station Address	72	24	0	16	0	0	0	[None]	[None]	[None]	nex
4	Type	96	16	0	16	0	0	0	[None]	Fig 4d	[None]	nex

Figure 4a

Id. at Fig. 4A; 29:32-30:4. Accordingly, the NumBits attribute of the Ethernet control record and the fdwoff attribute of the Ethernet Type field sub-record are the one or more locations in the packet where information is stored related to any child protocol of the particular protocol. *See also* Elements 1(b)(i)(1)-(2), 1(b)(iii)

GP: The GP PDF NumBits attribute of the protocol control record is set to 160, indicating the total bit length of the GP header:

GP Control Record					
Protocol Name	NumBits	NumFields	CurField	Fields	Options
GP - Generic Protocol	160	11	0	Figure 5a	Figure 6

Figure 5

Id. at Fig. 5.

The GP PDF "Frame Type," "Source Socket," and "Destination Socket" field sub-records each include a "Byte Offset" (i.e., fdwoff attribute), indicating the respective offsets from the start of the GP header:

Index	Field Name	Byte Offset	Bit Length	Left Shift	Right Shift	Check sum	Frame Length	Header Length	Statistics	Lookup Structure	Filter	Format
0	Version No.	0	4	0	28	0	0	0	Cnt/Index&Cnt	(None)	(None)	decimal
1	HeaderLen	0	4	4	28	0	0	32	Sum/Index&Cnt	Fig 5b	(None)	decimal
2	Frame Length	0	16	8	16	0	8	0	Sum	(None)	(None)	decimal
3	Frame Type	0	8	24	24	0	0	0	Index&Cnt	Fig 5c	Fig 10.Idx 2	hex
4	Checksum	4	16	0	16	ptr	0	0	(None)	(None)	(None)	hex
5	Control	4	8	16	24	0	0	0	(None)	(None)	(None)	bitfield
6	Hop Count	4	8	24	24	0	0	0	(None)	(None)	(None)	decimal
7	Source Socket	8	16	0	16	0	0	0	(None)	Fig 5d	(None)	hex
8	Destination Socket	8	16	16	16	0	0	0	(None)	Fig 5e	(None)	hex
9	Source Address	12	32	0	0	0	0	0	(None)	(None)	(None)	hex
10	Destination Address	16	32	0	0	0	0	0	(None)	(None)	(None)	hex

Figure 5a

Id. at Fig. 5A. Accordingly, the NumBits attribute of the GP control record and the fdwoff attribute of the GP Frame Type, Source Socket, and Destination Socket field sub-records are the one or more locations in the packet where information is stored related to any child protocol of the particular protocol. *See also* Elements 1(b)(i)(1)-(2), 1(b)(iii).

[1(b)(iii)] (iii) if there is at least one protocol specific operation to be performed on the packet for the particular protocol at the particular layer level, the one or more protocol specific operations to be performed on the packet for the particular protocol at the particular layer level; and

"Protocol specific operations" in accordance with the claim at least include "one or more parsing and extraction operations on the packet" (*see* Element 10(d)). The '725 Patent discloses: "[T]he PDL file for an Ethernet packet includes information on how the parsing subsystem is to extract the source and destination addresses, including where the locations and sizes of those addresses are." EX1033 at 42:3-6; *also* 41:57-65.

The Baker PDFs include the locations and sizes of fields that are to be extracted by the parsing control logic (e.g., the disclosed ParseFrame, ParseProtocol, ParseFields, ValidateValue, and GetValue control logic), among other information. *See generally* EX1038 at 19:11-20:34, Tables, 1-13, Figs. 4, 4A-D, 5, 5A-D (describing data written to PDF file); 26:26-40:36, Figs. 11-16 (describing control logic), 64:1-68:24; EX1006, ¶¶157-177 (describing parsing control logic as implemented by disclosed source code). Namely, each PDF field sub-record includes an "fdwoff" attribute (also called Byte/Bit Offset) that is the byte offset of the respective field from start of protocol header (i.e., the location of the field that is to be extracted). EX1038 at Table 2 ("fdwoff" attribute), 20:4, 27:17-35, Figs. 4A, 5A, 13 (step 210), 16 (step 502), 147:28-148:13, 148:25-26; EX1006, ¶¶169, 185, 149-150; *see* Element 1(b)(ii). Each PDF field sub-record also includes an "fblen" attribute (also called Bit Length) that is the length of the respective field in bits (i.e., the size of the field that is to be extracted). EX1038 at Table 2 ("fblen" attribute), 20:4, 35:27-30, Figs. 4A, 5A, 13 (step 222), 147:28-148:13, 148:25-26; EX1006, ¶¶185, 149-150. *See also* Element 1(c).

Each PDF also includes information on where in the packet to look for any child protocols. *See* Elements 1(b)(i)(1), 1(b)(ii), 1(c).

[1(c)] performing the protocol specific operations on the packet specified by the set of protocol descriptions based on the base protocol of the packet and the children of the protocols used in the packet,

The '725 Patent discloses "the base protocol" of a packet is the protocol associated with the packet type, such as Ethernet. EX1033 at column 71 (setting virtualChildren FIELD as "ethernet(1)"; disclosing "now define the base for the children" and setting VirtualBase PROTOCOL as "{VirtualChildren=virtualChildren}"), 32:29-33 ("At a base level, there are a number of packet types used in digital telecommunications, including Ethernet …."), 33:16-20 (FIG. 16 shows the header 1600 (base level 1) of a complete Ethernet frame (i.e., packet) …."), 32:9-12, 35:56-65, 40:1-22, 42:3-11; EX1006, ¶158.

As described in detail below, the Baker control logic performs protocol specific operations on the packet as specified by the set of protocol description files based on the base layer protocol (e.g., Ethernet) of the packet and the children of the protocols used in the packet (e.g., GP and others).

According to Baker, "once the system has received a network frame (at 100)," the base protocol of the packet (e.g., Ethernet) is determined by setting the ProtPtr variable (also referred to as the CurrentProtocol variable) to "the initial protocol description structure of the receiving interface number which for frame (2) is the Ethernet Protocol description defined in FIGS. 4-4d." EX1038 at 37:17-36;

also 67:6-21 (ParseFrame2 function receives ProtPtr as input), Fig. 11, Table 14 (CurrentProtocol, SrcIntf and IntTypes variables), 27:17-20, 52:12; EX1006, ¶¶158-160. The ParseFrame control logic "systematically parses through each network frame (at 104 to 108 and 128) until all known protocol headers have been parsed." EX1038 at 36:32-34; *also* 67:6-68:24 (ParseFrame2 function), Fig. 11-13; EX1006, ¶¶157-162. The ParseFrame control logic uses the ParseFields control logic to parse protocol fields as provided in accordance with protocol description records, starting with the initial or base protocol description record (e.g., Ethernet). EX1038 at 37:17-39:2, Figs. 11-13, EX1006, ¶¶157-177.

The ParseFields control logic "parses the fields in each protocol header" in accordance with Fig. 13 (30:10-31:2) and at step 210 initiates the GetValue control logic (implemented as get_value function in source code) to search for and locate a field in the packet and extract a value from the field. EX1038 at 27:17-35, 145:20-146:2, Figs. 13, 16; EX1006, ¶¶166-169. Namely, the GetValue control logic uses the fdwoff attribute (*see* Elements 1(b)(ii)-(iii)) as defined in the associated PDF and stored in an indexed element of a field array to locate and extract a value from a field (i.e., perform protocol specific operations specified by protocol descriptions):

> *"Referring now to FIG. 13, a value is extracted by the GetValue control logic (shown in FIG. 16) of the system (at 210) for each configured field that is relevant to the current network frame. As shown in FIG. 16, <u>the fdwoff</u>*

value is added to ParsePtr to locate and extract a 32-bit value (at 502) containing the field value[.] . . . The extracted value is returned (at 508) by the GetValue control logic."

EX1038 at 27:17-35.

```
U32 get_value()
  {
  U32 *ptr = (U32 *)(ParsePtr + fdwoff);
  U32 val = *ptr;
  if (fswap != 0)
    {
    val = wordswap(val);
    if (ptr2vary == 0)
      return((val << fshl) >> fshr);
    else   // This field needs its value varied
      return(*ptr = wordswap(ptr2vary->vary_value(val)));
    }
  else
    {
    if (ptr2vary == 0)
      return((val << fshl) >> fshr);
    else   // This field needs its value varied
      return(*ptr = ptr2vary->vary_value(val));
    }
  }
```

EX1038 at 145:20-146:2 (get_value function adds fdwoff variable (byte offset of the field) to ParsePtr variable (position of the start of the frame) to obtain location of the field in packet and extracts and returns value from packet); *also* 64:18, 148:26, Fig. 13 (step 210), Fig. 16 (step 502, "set ptr to 32 bit field at offset

(ParsePtr + fdwoff) set value to 32 bit value at offset"), Table 2; EX1006, ¶169; *see also* Element 1(d)(i).

At step 214 of Fig. 13, the ParseFields control logic further initiates the ValidateValue control logic (implemented as value_ok function in source code), which determines whether a lookup structure exists for a particular field and, if so, uses the Protocol attribute (implemented as "Prot" element in source code) of the associated lookup structure to "specify NextProtocol, the protocol description (at 308) to be used after current protocol header parsing is completed[.]." EX1038 at 29:1-31, Fig. 14; *also* 175:35-176:24, 64:25-38, 146:19-25; EX1006, ¶¶170-174. For example, when parsing the Ethernet Type field in accordance with the Ethernet PDF (Figs. 4-4D), the ValidateValue logic determines the next layer (i.e., child) protocol by comparing the extracted value from the Ethernet Type field (i.e., 0x8888) with the protocols present in the Fig. 4D lookup structure and sets the NextProtocol variable to the associated protocol description (Fig. 5) of the found value (i.e., GP):

> *"Using value 0×8888 as an example, if the ValidateValue control logic is applied to the Ethernet Type field and associated lookup structure shown in FIGS. 4a and 4d respectively, the lookup structure would be found (at 302), the value will be found in it (at 304), the associated Protocol field found with the range containing 0×8888 value will be used to update the NextProtocol variable (at 308) if it is NULL (at 306), and the associated Next Index field will be used to update the CurField variable."*

EX1038 at 29:32-30-4; *see also* Tables 2 ("ptr2np" attribute), 4 ("Protocol" attribute), 15:10-17, 29:1-31, Figs. 4A (Type field referencing Fig. 4D), 4D (GP Protocol identified by 0x8888 value and referencing Fig. 5 protocol description), 13-14. *See* Baker applied to Elements 1(d)(iii)-(iv). In this way, the ParseFields, GetValue, and ValidateValue control logic determine any child protocols of the protocol being parsed as specified by the associated PDF (i.e., perform protocol specific operations specified by protocol descriptions).

As the ParseFields control logic parses fields of a protocol header, a ProtoParseLen variable is updated by adding the BitLength/fblen attribute (*see* Element 1(b)(iii)) of each parsed field to the running value of the ProtoParseLen variable. EX1038 at 35:27-30, 65:20-24, Figs. 13 (step 222), 4A, 5A, Tables 2, 14; EX1006, ¶175. When parsing of the current protocol terminates, the ParsePtr variable is set to the start of the next layer protocol header/data (i.e., by adding the value of ProtoParseLen (which for Ethernet is equal to the value of NumBits) to the ParsePtr variable). EX1038 at 36:13-27, 64:11-13, 38:13-18, Figs. 4 ("NumBits" attribute), 13 (step 228); EX1006, ¶¶174, 161, 165.

Accordingly, Baker discloses that after the Ethernet protocol fields are parsed (at step 106 of Fig. 11), the NextProtocol variable "will refer to the GP shown in FIGS. 5-5(e), and ParsePtr will point at the start of line 2" of the frame. EX1038 at 38:13-18; *also* 146:19-25, 67:27-32, 29:17-31, Figs. 11-14, EX1006,

¶¶174, 161, 165. At this point, the CurrentProtocol variable will be updated with the NextProtocol value (of GP) and the GP fields in the frame are parsed using the ParseProtocol, ParseFields, GetValue, and ValidateValue fields described above. EX1038 at 38:18-24, 67:27-32, Fig. 11 (steps 108, 128); EX1006, ¶¶161-177. In one embodiment, once the GP fields have been parsed as specified by the GP protocol description, the Data fields will be parsed as specified by the associated protocol description (PDF):

> *"Referring back to the ParseFields control logic shown in FIG. 13, the ParseFields control logic parses the fields in each protocol header contained in a particular network frame by using field values obtained in accordance with information specified in associated protocol descriptions. The ParseFields control logic is applied for each protocol description required for a particular network frame. If the ParseFields control logic were applied to the exemplary frame, "Frame (1)," described above, the network interface system 10 of the present invention would apply the ParseFields control logic with the protocol descriptions for the Ethernet protocol shown in Table 12, the GP shown in Table 13, and an unspecified Data protocol description."*

EX1038 at 30:10-23; *also* 33:30-32 ("The statistics entity of the field sub-record may be used to indicate categories of statistics to be maintained for each protocol header field (at 218 and 236)."), 36:28-36, 67:33-36; *see generally* 3:32-5:9, 5:4-9, 26:26-30:35, 33:21-34-7, 35:13-38:24, 19:3-23-8, Tables 1-4, 12-14, Figs. 4-5D, 11-14; EX1006, ¶162.

[1(d)(i)] the method further comprising:

storing a database in a memory, the database generated from the set of protocol descriptions and including a data structure containing information on the possible protocols and organized for locating the child protocol related information for any protocol,

According to the '725 Patent, a database may include a Protocol Table (PT) and a series of one or more Look Up Tables (LUTs) associated with a protocol in PT and used to identify known protocols and their children. EX1033 at 14:39-43, 39:22-40:60, Fig. 18B. Baker discloses a database in memory generated from the set of protocol descriptions consistent with the '725 Patent Fig. 18B embodiment. EX1006, ¶¶182-187. Namely, Baker discloses that, upon initialization and when PDF files are read into memory, the resulting protocol description data structures include a data structure of protocol records (each corresponding to a PDF) and further provides a ProtocolList sorted vector that is used as an index for searching the protocol descriptions supported by the system. EX1038 at 20:35-21:11, 127:20-128:13, 128:33-129:1; EX1006, ¶¶143-147, 183. The protocol data structure includes the elements depicted in Table 1, including an indexed field array of one or more field sub-records:

```
U32     name_length;   // Length of protocol name in bytes
S8      *protocol_name; // Protocol name in Ascii
S8      *fname;        // File name in Ascii
```

```
U32       num_bits;      // # of bits in fixed portion of protocol header
U16       num_fields;    // # of fields used to describe protocol header
U16       cur_field;     // Index of last field displayed
U32       out_flag;      // Protocol has been output 'flag'
U32       dbW;           // Display bit width (i.e. #bits per line)
field     *fs;           // Fields for fixed portion of protocol header
protocol  *opts;         // Pointer to protocol for parsing options
```

EX1038 at 155:35-154:8 (elements of protocol data structure) *also* Table 1; EX1006, ¶146.

```
fs = new field[num_fields];
for (U32 i=0; i<num_fields; i++)
  fs[i].get_from_file(fp);
```

EX1038 at 57:7-9 (allocation of field array to size of num_fields element of PDF; records stored in array starting at index 0 and incrementing to num_fields); EX1006, ¶148-150.

Each field entry in the field sub-record array includes the elements depicted in Table 2 and may include a ptr2np variable that, when set, is a pointer to a next protocol lookup structure.

```
U32     fdwoff;   // field offset in bytes
U8      fshl;     // Number of bits to shift left
U8      fshr;     // Number of bits to shift right
U8      ffmt;     // Field output display format
U8      flflag;   // if TRUE this field contains length of option
U8      reserved; // not used...pad to align following fields
U8      fmult;    // multiply value by this amount before display
U8      fswap;    // Flag indicating the need to swap bytes and words
U8      fdspfield; // display this field on output
S8      *fname;   // user defined field name
stats   *ptr2stats; // Pointer to derived statistics class
lookup  *ptr2np;  // Pointer to derived lookup class (next_protocol)
vary    *ptr2vary; // Pointer to Vary field value class
CheckSum *ptr2csum; // Pointer to Checksum class for current protocol
criteria *ptr2flt; // Pointer to Filter Criteria class for this field
```

EX1038 at 148:24-149:7; *also* Table 2; EX1006, ¶149. The field array also includes an "fdwoff" variable describing the "field offset in bytes" and an "fblen" variable describing "field length in bits." *Id.* The fdwoff variable is used to determine the location of the field in a packet that may relate to the child protocol. EX1038 at 145:20-146:2, 27:17-35, Figs. 13 (step 210), 16; EX1006, ¶169.

For multi-protocol next protocol lookups, each next protocol lookup structure is an array of next protocol records, where each entry in the array includes the elements depicted in Table 4 and is indexed by a value that is extracted from the packet and used to determine the child protocol.

```
// Setup lookup from file
   void get_from_file(FILE *fp)
      {
      U32 num;
      fread(&num, sizeof(num), 1, fp);      // Write out number of valid array entries
      for (U32 j=0; j<num; j++)
         {
         verify v;
         v.get_from_file(fp);
         v.maxval = min(v.maxval, size);
         for (U32 i=v.minval; i<=v.maxval; i++)
            {
            if (((i & 1) && (v.okbits & 1)) || ((i & 1) == 0 && v.okbits > 1))
               {
               array[i].prot = v.prot;
               array[i].nxtidx = v.nxtidx;
               array[i].minval = i;
               array[i].maxval = i;
               array[i].okbits = v.okbits;
               }
            }
         }
      }
```

EX1038 at 177:4-26; *also* 148:11-12, 165:37-166:5, 171:23-177:33, Table 4; EX1006, ¶¶150-156.

During parsing, the protocol data structure is used to extract a field that specifies the next protocol, which is then used as an address into the lookup structure for determining the child protocol. EX1038 at 145:20-146:2, 64:18, 148:26, 27:17-35, Fig. 13 (step 210), Fig. 16, Table 2 (get_value function determines location of field in packet and extracts value from it), EX1006, ¶169, 186 *and* 64:25-38, 175:35-176:24, 29:1-31, Fig. 13 (step 214), Fig. 14 (value_ok

function uses extracted value as index to lookup structure to determine if value is for a valid protocol), EX1006, ¶¶170-174, 186. When the next protocol is identified using the lookup structure, it is used as an index into the protocol table. EX1038 at 146:19-25, 29:17-31, 67:27-32, Fig. 11 (steps 108, 128), Fig. 14 (steps 306, 308, 310, 312) (if the extracted value is found in a lookup structure, set pointer to next protocol's data structure); EX1006, ¶¶174, 161, 186. In light of the above, and as explained by a POSITA, Baker discloses this limitation. EX1006, ¶¶182-188.

[1(d)(ii)] the data structure contents indexed by a set of one or more indices,

Baker discloses that multi-protocol lookup structures are each indexed by a value extracted from the next/child protocol field in the packet. EX1038 at 177:4-26 (reproduced above); *also* 64:25-38, 175:35-176:24, 29:1-31, Fig. 13 (step 214), Fig. 14 (value_ok function uses extracted value as index to lookup structure to determine if value is for a valid protocol); EX1006, ¶¶151-156, 170-172. The '725 Patent similarly discloses that each LUT is "indexed by one byte of the child recognition pattern that is extracted from the next protocol field in the packet." EX1033 at 39:33-35.

When the next/child protocol is identified using the Baker lookup structure, it is used as an index into the protocol table. EX1038 at 146:19-25, 29:17-31, 67:27-32, Fig. 11 (steps 108, 128), Fig. 14 (steps 306, 308, 310, 312), Table 4 (if

the extracted value is found in a lookup structure, set pointer to next protocol's data structure); EX1006, ¶¶174, 161, 190. The '725 Patent similarly discloses that when a lookup results in a valid next protocol, the next protocol "is used as an index into the protocol table." EX1033 at 40:27-31.

Baker also discloses a ProtocolList sorted vector that is used as an index for searching protocol descriptions supported by the system. EX1038 at 127:20-128:13, 128:33-129:1; EX1006, ¶¶143-147, 191. And, as discussed above, each protocol record includes an indexed array of field sub-records. EX1038 at 147:28-148:13 and 148:24-149:7 (reproduced above), Table 2; EX1006, ¶¶148-150, 191. The field array index is used "to determine where parsing of the current protocol will continue after the current field." EX1038 at 16:47-53; *also* Fig. 4D, 5C-5E (Next Index), 16:60-67, 64:11-13, 36:13-17, Table 4; EX1006, ¶¶176-177, 191. In light of the above, and as explained by a POSITA, Baker discloses this limitation. EX1006, ¶¶189-192.

[1(d)(iii)] the database entry indexed by a particular set of index values including an indication of validity,

According to the '725 Patent, each LUT entry includes a "node code" that indicates the validity of the contents. EX1033 at 39:39-50, 35:4-11, 14:57-61. A "protocol" node code indicates a recognized protocol. *Id.* A "null" node code indicates "that there is no valid entry." *Id.*

As discussed above, the Baker multi-protocol lookup structure entries are indexed by a valid protocol identification value. *See* Elements 1(d)(i)-(ii) above; *also* EX1006, ¶¶151-156, 170-172, 194. Entries also include a "prot" variable ("Protocol" of Table 4 and Figs. 4D, 5C-5E) that, similar to the '725 Patent "node code," has one of two values: (1) a pointer to a protocol record, indicating the protocol that has been recognized as the next/child protocol; or (2) a "null" code indicating an "unknown" or "illegal" entry (i.e., there is no valid entry):

Frame Type Next Protocol Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	4	0x00	0x00	ALL	"Illegal Protocol"
GP1	4	0x01	0x01	ALL	"GP1"
GP2	5	0x02	0x02	ALL	"GP2"
[None]	4	0x03	0xFF	ALL	"Illegal Protocol"

Fig. 5C

EX1038 at Fig. 5C.

Source Socket Next Protocol Structure					
Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	8	0x0000	0x0142	ALL	"Unknown Protocol"
GP3	8	0x0143	0x018F	ODD	"GP3"
GP4	8	0x0143	0x018F	EVEN	"GP4"
[None]	8	0x0190	0xFFFF	ALL	"Illegal Protocol"

Fig. 5D

Id. at Fig. 5D; *also* Fig. 5E, Table 4 (describing "Protocol" variable as "pointer to protocol description structure"); *see* Element 1(d)(i); EX1006, ¶195.

During parsing, the value_ok function (also called the ValidateValue control logic) uses an extracted value as an index to the lookup structure to determine if the value is for a valid protocol.

> *"Values or ranges of values found in configured lookup structures are considered to be valid. The Prot and NextIndex values associated with a value or range found are used to specify NextProtocol, the protocol description (at 308) to be used after current protocol header parsing is completed, and the index of the next field (at 310) is used to determine where parsing of the current protocol will continue after the current field. ...*
>
> *The ValidateValue control logic returns an updated CurField value (at 312 and 316) together with a valid/invalid indication, and where indicated (at 308) may return an updated value for NextProtocol."*

EX1038 at 29:17-31.

```
verify *value_ok(U32 value)
  {
```

```
if (value > size)
   return(0);
else
   {
   verify *v = &array[value];
   if (v->okbits != 0)
      {
      if (value >= v->minval && value <= v->maxval)
         {
         if (value & 1) // It's an odd number
            {
            if (v->okbits & 1)          // ODD and ALL numbers ok
               return(v);
            }
         else          // It's an even number
            {
            if (v->okbits > 1)    // EVEN and ALL numbers ok
               return(v);
            }
         }
      }
   return(0);     // Value was not found
   }
}
```

EX1038 at 175:35-176:24 (value_ok sub-function searches lookup array at the indexed location corresponding to the extracted value (array[value]), tests whether value is valid entry, and returns a valid/invalid indication to main value_ok function); *also* 64:25-38, 29:1-31, Fig. 13 (step 214), Fig. 14; EX1006, ¶¶170-172.

```
else
  {
  if (tp == 0)
    tp = v->prot;
  i = v->nxtidx;
  return(TRUE);
  }
```

EX1038 at 146:19-25 (if valid entry indication returned from value_ok sub-function, main value_ok function sets local next protocol variable to "prot" variable of the valid entry and returns valid indication); *also* Fig. 14 (steps 306, 308, 310, 312); EX1006, ¶174. In light of the above, and as explained by a POSITA, Baker discloses this limitation. EX1006, ¶¶193-196.

[1(d)(iv)] wherein the child protocol related information includes a child recognition pattern,

Baker's "Generic Protocol" (GP) defines three fields of a packet (Frame Type, Source Socket, Destination Socket) that may lead to identification of the child protocol:

Index	Field Name	Byte Offset	Bit Length	Left Shift	Right Shift	Check sum	Frame Length	Header Length	Statistics	Lookup Structure	Filter	Format
0	Version No.	0	4	0	28	0	0	0	Cnt/Index&Cnt	[None]	[None]	decimal
1	HeaderLen	0	4	4	28	0	0	32	Sum/Index&Cnt	Fig 5b	[None]	decimal
2	Frame Length	0	16	8	16	0	8	0	Sum	[None]	[None]	decimal
3	Frame Type	0	8	24	24	0	0	0	Index&Cnt	Fig 5c	Fig 10.Idx 2	hex
4	Checksum	4	16	0	16	ptr	0	0	[None]	[None]	[None]	hex
5	Control	4	8	16	24	0	0	0	[None]	[None]	[None]	bitfield
6	Hop Count	4	8	24	24	0	0	0	[None]	[None]	[None]	decimal
7	Source Socket	8	16	0	16	0	0	0	[None]	Fig 5d	[None]	hex
8	Destination Socket	8	16	16	16	0	0	0	[None]	Fig 5e	[None]	hex
9	Source Address	12	32	0	0	0	0	0	[None]	[None]	[None]	hex
10	Destination Address	16	32	0	0	0	0	0	[None]	[None]	[None]	hex

Figure 5a

EX1038 at Fig. 5A; *see also* Element 1(b)(i)-(ii). When stored in the field sub-record array, each of these three field records includes a ptr2np pointer to an associated next protocol lookup array (depicted in Figs. 5C-5E). *See* Element 1(d)(i); EX1006, ¶¶198-199. Each of these lookup structures includes multiple protocol entries over a range of corresponding valid values for the associated protocol. *See, e.g.,*

Frame Type Next Protocol Structure

Protocol	Next Index	Minimum	Maximum	Mask	Translation
[None]	4	0x00	0x00	ALL	"Illegal Protocol"
GP1	4	0x01	0x01	ALL	"GP1"
GP2	5	0x02	0x02	ALL	"GP2"
[None]	4	0x03	0xFF	ALL	"Illegal Protocol"

Figure 5c

EX1038 at Fig. 5C; *also* Figs. 5D-5E; EX1006, ¶199. Each lookup structure is stored as an array, indexed by a valid protocol identification value (i.e., child recognition pattern). *See* Element 1(d)(i); EX1006, ¶¶151-156, 199. For example, as depicted in Fig. 5C, Baker discloses that the child protocol (e.g., GP1) of a particular protocol (e.g., GP), is indicated by a child recognition pattern (e.g., 0x01 (hex)). *Compare* EX1038 at Fig. 5C-5E *with* EX1033 at 35:23-30; EX1006, ¶200. In light of the above, and as explained by a POSITA, Baker discloses this limitation. EX1006, ¶¶197-202.

[1(e)] wherein step (c) of performing the protocol specific operations includes, at any particular protocol layer level starting from the base level, searching

the packet at the particular protocol for the child field, the searching including indexing the data structure until a valid entry is found,

Baker discloses performing protocol specific operations starting from the "base" level (e.g., Ethernet). *See* Element 1(c). Such operations include, at any level, searching the packet for fields that include information on the child protocol. Namely, Baker discloses ParseFields control logic, that iterates through fields of a packet that are defined in an associated indexed field array of a particular protocol description record in accordance with Fig. 13. EX1038 at Fig. 13; EX1006, ¶¶166-168. For each field that is to be parsed in accordance with the protocol description and starting at index (i=0) of the field sub-records array, the ParseFields control logic initiates the get_value function (also called the GetValue control logic) to determine the location of the field in the packet and extract a value from the field. EX1038 at 64:11-18, EX1006, ¶¶168-169; *See* Element 1(c). If the field includes a child recognition pattern (*see* Element 1(d)(iv)), Baker initiates the value_ok function (also called the ValidateValue control logic) to use the extracted value as an index to the associated lookup array and determine if the value is for a valid protocol. *See* Element 1(d)(iii). If the currently parsed field either doesn't contain a child recognition pattern or the value_ok function determines that there is not a valid entry for the child recognition pattern, the index used to increment through the field array is incremented by 1 and the ParseFields logic continues by parsing

the next field of the field array. EX1038 at 146:9-18, 64:11-13, 29:7-16, Figs. 13, 14; EX1006, ¶¶170-172.

If at any point, the currently parsed field includes a child recognition pattern and the value_ok function determines that it is associated with a valid child protocol, the NextProtocol variable is set to the associated protocol description that will be used to parse the next layer of the packet for the "first valid field parsed in a protocol that specifies the NextProtocol." EX1038 at 146:19-25, 29:17-31, Fig. 14; EX1006, ¶¶172-174. The ParseFields logic continues implementing the get_value and value_ok functions in this way until a valid child protocol is identified or when processing otherwise is terminated, at which point, parsing of the next layer protocol continues in accordance with the identified NextProtocol protocol description using the ParseFields logic. EX1038 at 64:11-13, 36:13-17, Figs. 11, 13; EX1006, ¶¶176, 168-175, 161-165. In this way, the Baker control logic "systematically parses through each network frame (at 104 to 108 and 128) until all known protocol headers have been parsed" at which point "[a]ny remaining frame bits are parsed as application data (at 110, 112 and 130) and/or pad data (at 114, 116 and 132)." EX1038 at 36:32-36; *also* Figs. 11-16; EX1006, ¶¶162.

Accordingly, Baker discloses searching the packet at the particular protocol for a field that includes a child recognition pattern (i.e., the child field). This

search includes indexing until a valid entry is found at least because fields of the protocol are searched by searching the packet for the field indicated by the indexed field array entry, and incrementing the field array index until the field contains a child recognition pattern (i.e., until a field includes a valid "ptr2np" entry). EX1006, ¶¶204-205, 176.

Baker also discloses searching the packet at the particular protocol for a field that includes a valid child recognition pattern (i.e., the child field). This search includes indexing until a valid entry is found at least because, for multi-protocol lookups, Baker uses an extracted child recognition pattern as an index to the associated lookup array and determines if the child recognition pattern is valid. EX1006, ¶¶170-172, 190, 206; *see also* Element 1(d)(iv). Searching (i.e., incrementing the field array index, extracting value, determining validity) continues until a valid child recognition pattern is found. EX1006, ¶206.

[1(f)] and whereby the data structure is configured for rapid searches using the index set.

It is Petitioners position that the data structure being configured for "rapid" searches is not entitled to any patentable weight because it claims a desired a result. *See* Section II.B.2.d. Baker discloses a data structure configured for searches using the index set. *See* Elements 1(e), 1(d)(ii). However, if "rapid" searches is deemed to be limiting, a POSITA understands that Baker discloses the same within the

meaning of the '725 Patent. EX1006, ¶¶209-210. Namely, the '725 patent discloses "rapid searches using the index set" in the following context:

> *"An alternate embodiment of the data structure used in database 308 is illustrated in FIG. 18B. [T]he data structure permits rapid searches to be performed by the pattern recognition process 304 by indexing locations in a memory rather than performing address link computations. ...*
>
> *The pattern matching is carried out by finding particular "child recognition codes" in the header fields, and using these codes to index one or more of the LUT's."*

EX1033 at 14:33-62. The Baker data structure uses a child recognition code/pattern (*see* Element 1(d)(iv)) to index multi-protocol next protocol lookup arrays in the exact same way. EX1038 at 177:4-26; EX1006, ¶210. Additionally, searching the Baker multi-protocol lookup arrays is performed by indexing locations in memory rather than performing address link computations. EX1038 at 64:25-38, 175:35-176:24, 29:1-31, Fig. 13 (step 214), Fig. 14 (value_ok function uses extracted value as index to lookup structure to determine if value is for a valid protocol); EX1006, ¶¶170-172, 210. Searching the Baker multi-protocol lookup arrays is the same as searching the Lookup Tables of the '725 Patent. *Compare* Baker as discussed above *with* EX1033 at 14:33-62; EX1006, ¶¶209-210. Accordingly, searching the Baker multi-protocol lookup arrays by indexing locations in memory is a "rapid search" using the child recognition pattern index set within the meaning of the '725 patent. EX1006, ¶¶209-210.

Claim 2. A method according to claim 1, wherein the protocol descriptions are provided in a protocol description language, the method further comprising: compiling the PDL descriptions to produce the database.

Each of Baker's PDF files describes a protocol that may be used in packets encountered by the monitor. EX1038, *Baker* at 11:22-25 ("[E]ach of these protocol description records with its associated field, statistics, lookup, and filter record information is also written to a protocol specific protocol description file (PDF)."); EX1006, ¶213. Each PDF describes commands for determining the length of the header. *Compare* EX1033 at 48:41-50 (describing "HEADER" attribute), column 73 ("HEADER { LENGTH=14 }") *with* EX1038 at Table 1 ("numBits" attribute), 11:32 (describing "numbBits" attribute as "the total bit length of the protocol header"), 57:1; EX1006, ¶214. Each PDF describes commands for defining the fields within the protocol header. *Compare* EX1033 at 47:34-48:18 (describing "PROTOCOL" definition), column 79 (PROTOCOL section provides "Detailed packet layout for the IP datagram. This includes all fields and format. All offsets are relative to the beginning of the header.") *with* EX1038 at Table 1 ("fields" attribute), Table 2 ("fblen" and "fdwoff" attributes), 11:49-50 (describing "fblen" attribute as "the length of the field in bits" and "fdwoff" attribute as "the byte offset from the start of protocol header"), 11:36-40, 147:35-36; EX1006, ¶215. Further, each PDF describes commands for determining the protocols used at the next layer. *Compare* EX1033 at 49:45-55 ("CHILDREN" attribute used to "used

to describe how children protocols are determined."), column 79 ("CHILDREN { DESTINATION=Protocol }") *with* EX1038 at Table 2 ("ptr2np" attribute includes a "pointer to lookup structure/class … next protocol definition to use (0 = none)"), Table 4 ("Protocol" attribute describing "pointer to protocol description"), 8:13-15, 148:10-12, 165:37-166:5, 177:4-26; EX1006, ¶216. A POSITA recognizes that Baker's PDFs are written in a protocol description language. EX1006, ¶¶212-217

Consistent with the compilation process disclosed in the '725 Patent, Baker discloses implementation of the system on a network device capable of storing programmably configurable PDFs and programmed to run an initialization/compilation process to produce the searchable and indexed database discussed throughout claim 1:

> *"[T]he initialization of the system includes a determination of the presence of PDF files and the extraction of the protocol and associated control record information from all of the PDF files found. The number of PDF files is determined, and a ProtocolList is constructed consisting of a sorted vector of protocol records at least the size of the number of PDF files found. The name of each protocol record found is inserted in the ProtocolList. The PDF files are then read to memory in the sequence described above for the PDF file writing. The lookup records that indicate a next protocol are associated with the appropriate entries in the ProtocolList."*

EX1038 at 12:7-18; *also* 2:63-3:7, 11:26-12:6; EX1006, ¶¶143-156, 219; *see also* EX1033 at 9:22-27 ("Both the pattern information for parsing and the related extraction operations, e.g., extraction masks, are supplied from a parsing-pattern-structures and extraction-operations database (parsing/extractions database) 308 filled by the compiler and optimizer 310."), 9:42-44, 9:53-54, 22:7-10, 13:53-57, 14:63-15:17, Fig. 4.

V. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8(A)(1)

A. Real Party-In-Interest and Related Matters

Petitioners are the real parties-in-interest. 37 C.F.R. § 42.8(b)(1). The '725 Patent is the subject of a lawsuit filed by Patent Owner against Petitioners in the U.S. District Court for the Eastern District of Texas. EX1019; 37 C.F.R. § 42.8(b)(2). The '725 Patent is also the subject of one other pending lawsuit and numerous terminated lawsuits, a full list of which can be found at EX1020. 37 C.F.R. § 42.8(b)(2). In addition, Petitioners have filed IPRs on the related '646, '751, '789, and '099 Patents. *See* IPR2017-00450, IPR2017-00451, IPR2017-00629, IPR2017-00630, IPR2017-00769. Petitioners are concurrently filing a petition for IPR on additional claims of the '725 Patent. *See* IPR2017-00862.

B. Lead and Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)

Petitioners provide the following designation and service information for lead and back-up counsel.

Eric A. Buresh (Reg. No. 50,394) (LEAD)

eric.buresh@eriseip.com
Mark C. Lang (Reg. No. 55,356) (Back-Up)
mark.lang@eriseip.com
Kathleen D. Fitterling (Reg. No. 62,950) (Back-Up)
kathleen.fitterling@eriseip.com
6201 College Blvd., Suite 300
Overland Park, Kansas 66211
Telephone: (913) 777-5600
Fax: (913) 777-5601

Abran J. Kean, Reg. No. 58,540 (Back-Up)
abran.kean@eriseip.com
5600 Greenwood Plaza Blvd., Suite 200
Greenwood Village, CO 80111
Phone: (913) 777-5600

C. Payment of Fees Under 37 C.F.R. § 42.103

The undersigned submitted payment by deposit account with the filing of this Petition authorizing the Office to charge fees required under 37 C.F.R. § 42.103(a) and 42.15(a).

VI. CONCLUSION

Petitioners respectfully request that claims 1-2 of the '725 Patent be canceled.

Date: February 9, 2017

Respectfully submitted,
ERISE IP, P.A.

BY: /s/ *Eric A. Buresh*/
Eric A. Buresh, Reg. No. 50,394
Mark C. Lang, Reg. No. 55,356
Kathleen D. Fitterling, Reg. No. 62,950
6201 College Blvd., Suite 300
Overland Park, KS 66211

P: (913) 777-5600
F: (913) 777-5601
eric.buresh@eriseip.com
mark.lang@eriseip.com
kathleen.fitterling@eriseip.com

Abran J. Kean, Reg. No. 58,540
5600 Greenwood Plaza Blvd., Suite 200
Greenwood Village, CO 80111
Phone: (913) 777-5600
abran.kean@eriseip.com

ATTORNEYS FOR PETITIONERS

APPENDIX OF EXHIBITS

EX1001	RESERVED
EX1002	U.S. Patent No. 6,839,751 ("the '751 Patent") to Dietz, et al., filed on June 30, 2000, and issued on January 4 ,2005
EX1003	RESERVED
EX1004	RESERVED
EX1005	U.S. Provisional App. No. 60/141,903, filed on June 30, 1999
EX1006	Expert Declaration of Bill Lin
EX1007	RESERVED
EX1008	RESERVED
EX1009	RESERVED
EX1010	RESERVED
EX1011	RESERVED
EX1012	RESERVED
EX1013	RESERVED
EX1014	RESERVED
EX1015	RESERVED
EX1016	RESERVED
EX1017	RESERVED
EX1018	RESERVED
EX1019	Complaint in *Packet Intelligence, LLC v. Sandvine Corporation, et al.,* Case No. 2:16-cv-147 (E.D. Tex.)
EX1020	Listing of Cases Involving Challenged Patent
EX1021	RFC 1067, "A Simple Network Management Protocol," published August 1988 by Network Working Group
EX1022	RFC 1271, "Remote Network Monitoring Management Information Base," published November 1991 by Network Working Group
EX1023	RFC 1757, "Remote Network Monitoring Management Information Base," published February 1995 by Network Working Group
EX1024	RFC 2012, "SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2," published November 1996 by Network Working Group
EX1025	RESERVED
EX1026	RESERVED

EX1027	RESERVED
EX1028	Harry Newton. 1996. Newton's Telecom Dictionary (11th ed.). Flatiron Publishing, Inc., Command Web, New Jersey, USA, pp. 48, 169.
EX1029	RESERVED
EX1030	RESERVED
EX1031	RESERVED
EX1032	RESERVED
EX1033	U.S. Patent No. 6,665,725 ("the '725 Patent") to Dietz, et al., filed on June 30, 2000, and issued on December 16, 2003
EX1034	RESERVED
EX1035	RESERVED
EX1036	RESERVED
EX1037	RESERVED
EX1038	International Publication No. WO 97/23076 to Baker, et al. ("Baker"), filed on December 13, 1996, and published on June 26, 1997, with corrected figures published on October 16, 1997, with corrected figures published on October 16, 1997, and claiming priority to U.S. Application Ser. No. 08/575,506 filed on December 20, 1995
EX1039	'725 Patent File History
EX1040	IEEE Std 802.3 – 2012, "IEEE Standard for Ethernet," published December 28, 2012, pp. 53-58
EX1041	IEEE Std 802.3x-1997 and IEEE Std 802.3y-1997, "IEEE Standards for Local and Metropolitan Area Networks: Supplements to Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications—Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 or Better Balanced Twisted Pair Cable (100BASE-T2)," approved March 20, 1997 by IEEE Standards Board and September 5, 1997 by American National Standards Institute, pp. 28-32
EX1042	RFC 894, "A Standard for the Transmission of IP Datagrams over Ethernet Networks," published April 1984 by Network Working Group
EX1043	RFC 1010, "Assigned Numbers," published May 1987 by Network Working Group

EX1044	RFC 791, "Internet Protocol DARPA Internet Program Protocol Specification," published September 1981
EX1045	RFC 793, "Transmission Control Protocol DARPA Internet Program Protocol Specification," published September 1981
EX1046	RFC 768, "User Datagram Protocol," published August 28, 1980
EX1047	RESERVED

CERTIFICATE OF COMPLIANCE

Pursuant to 37 C.F.R. § 42.24(d), I hereby certify that this Petition complies with the type-volume limitation of 37 C.F.R. § 42.24(a)(1)(i) because it contains 8,221 words as determined by the Microsoft® Office Word word-processing system used to prepare the brief, excluding the parts of the brief exempted by 37 C.F.R. § 42.24(a)(1).

/s/*Eric A. Buresh*/
Eric A. Buresh

CERTIFICATE OF SERVICE ON PATENT OWNER
UNDER 37 C.F.R. § 42.105(a)

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(b), the undersigned certifies that on February 9, 2017, a complete and entire copy of this Petition for *Inter Partes* Review was provided via Federal Express to the Patent Owner by serving the correspondence address of record for the '725 Patent and Patent Owner's litigation counsel:

Meunier Carlin & Curfman LLC
999 Peachtree Street NE
Suite 1300
Atlanta, GA 30309

Steven W. Hartsell
SKIERMONT DERBY LLP
2200 Ross Avenue, Suite 4800W
Dallas, Texas 75201

ERISE IP, P.A.

BY: /s/ *Mark C. Lang/*

Eric A. Buresh, Reg. No. 50,394
Mark C. Lang, Reg. No. 55,356
Kathleen D. Fitterling, Reg. No. 62,950
6201 College Blvd., Suite 300
Overland Park, KS 66211
P: (913) 777-5600
F: (913) 777-5601
eric.buresh@eriseip.com
mark.lang@eriseip.com
kathleen.fitterling@eriseip.com

Abran J. Kean, Reg. No. 58,540
5600 Greenwood Plaza Blvd., Suite 200
Greenwood Village, CO 80111
Phone: (913) 777-5600

ATTORNEYS FOR PETITIONERS