

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,

Petitioner,

v.

HEADWATER RESEARCH LLC,

Patent Owner.

Case No. IPR2026-00049

U.S. Patent No. 9,179,359

DECLARATION OF DR. KEVIN BUTLER

Table of Contents

I.	INTRODUCTION	1
II.	SUMMARY OF OPINIONS	1
III.	QUALIFICATIONS AND BACKGROUND	2
IV.	MATERIALS CONSIDERED	7
V.	LEGAL STANDARDS	8
	A. Claim Construction.....	9
	B. Anticipation Under 35 U.S.C. § 102	10
	C. Obviousness Under 35 U.S.C. § 103.....	10
	D. Secondary Considerations of Non-Obviousness.....	13
VI.	TECHNOLOGY OVERVIEW.....	15
	A. Mobile Devices.....	15
	B. Resource Management in Mobile Devices	19
VII.	THE '359 PATENT.....	20
	A. Specification.....	20
	B. Prosecution History	23
VIII.	CLAIM CONSTRUCTION	23
IX.	LEVEL OF ORDINARY SKILL	25
X.	OVERVIEW OF PRIOR ART	26
	A. Shell.....	26
	B. Cole.....	34
	C. Flack	37
XI.	GROUND 1: SHELL AND COLE RENDER OBVIOUS CLAIMS 1 AND 26.....	39

A.	Motivations to Combine.....	39
B.	Independent Claim 1	44
1.	[1pre]: “A wireless end-user device, comprising:”	44
2.	[1a]: “a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN;”	45
3.	[1b]: “a wireless local area network (WLAN) modem to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN; and”	49
4.	[1c]: “one or more processors configured to apply a stored WWAN differential traffic control policy to Internet data service provided using the WWAN modem and the at least one WWAN, and”	53
5.	[1d]: “indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic control policy,”	59
6.	[1e]: “wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”	69
C.	Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”	81
XII.	GROUND 2: SHELL, COLE, AND FLACK RENDER OBVIOUS CLAIM 1.....	85
1.	[1pre]-[1d]:.....	85

2. [1e]: “wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”86

XIII. GROUND 3: SHELL, COLE, AND FLACK RENDER OBVIOUS
CLAIM 2691

- A. Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”91

XIV. CONCLUSION.....96

I. INTRODUCTION

1. I have been retained on behalf of Google LLC (“Petitioner”) to offer technical opinions related to U.S. Patent No 9,179,359 (“the ’359 patent,” EX1001). I understand that Petitioner is requesting that the Patent Trial and Appeal Board (“PTAB” or “Board”) institute an inter partes review (“IPR”) proceeding of the ’359 patent.

2. I have been asked to provide my independent analysis of the ’359 patent based on the prior art publications cited in this declaration.

3. I am not and never have been an employee of Petitioner. I received no compensation for this declaration beyond my normal hourly compensation based on my time actually spent analyzing the ’359 patent, the prior art publications cited below, and issues related thereto, and I will not receive any added compensation based on the outcome of any IPR or other proceeding involving the ’359 patent.

II. SUMMARY OF OPINIONS

4. This declaration considers claims 1 and 26 (“Challenged Claims”) of the ’359 patent. I have been informed that Patent Owner disclaimed claim 1, but I should still consider claim 1 because claim 26 depends from it. Below, I set forth the opinions I have formed, the conclusions I have reached, and the bases for these opinions and conclusions. I believe the statements contained in this declaration to be true and correct to the best of my knowledge.

5. Based on my experience and knowledge of the art at the time of the applicable priority date, it is my opinion that the Challenged Claims would have been obvious to one of ordinary skill in the art based on the grounds below:

Ground	Claims Challenged	Basis	Reference(s)
1	1 and 26	§103	Shell and Cole
2	1	§103	Shell, Cole, and Flack
3	26	§103	Shell, Cole, and Flack

III. QUALIFICATIONS AND BACKGROUND

6. I am over the age of 18 and am competent to write this declaration. I have personal knowledge or have developed knowledge, based upon education, training, or experience, of the matters set forth herein.

7. I have summarized in this section my educational background, career history, publications, and other relevant qualifications.

8. I received a B.Sc. degree in Electrical Engineering from Queen's University in 1999, an M.S. degree in Electrical Engineering from Columbia University in 2003, and a Ph.D. in Computer Science and Engineering from the Pennsylvania State University in 2010.

9. I am a Professor with tenure in Computer and Information Science and Engineering and Director of the Florida Institute for Cybersecurity Research at the

University of Florida, where I hold a University of Florida Research Foundation Professorship. I founded and directed the Oregon Systems Infrastructure and Information Security (OSIRIS) Lab at the University of Oregon, where I was an assistant professor of computer and information science for four years before moving to Florida under the Rising to National Preeminence initiative. I have authored over 150 peer-reviewed publications including book chapters, journal articles, conference proceedings papers, and workshop papers. My research is in the area of computer systems security, including embedded systems and firmware, as well as network security, and with a particular focus on the security of peripheral devices that use the USB protocol and the hosts that these devices connect to. I received a National Science Foundation CAREER award in 2013, an International Educator of the Year award within the Herbert Wertheim College of Engineering at the University of Florida in 2017, a University of Florida Term Professorship for 2018-2021 and 2021-2024, and an Arnold and Lisa Goldberg Rising Star Endowed Professorship in Computer Science for 2018, which I relinquished in 2021 upon promotion to Full Professor. I also received an Outstanding Community Service Award from the IEEE Technical Committee on Security and Privacy in 2017. I am a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), Senior Member of the Association for Computing Machinery (ACM), and Fellow of the European Alliance for Innovation (EAI). I have been involved with international

standards-setting bodies; in particular, I was co-chair and leader of the Security Workstream for the International Telecommunications Union (ITU)'s Technology, Innovation, and Competition Working group within the Focus Group on Digital Financial Service from 2015-2016, and from 2017-2022, served as Co-Chair of the Security, Infrastructure, and Trust Working Group within the ITU's Financial Inclusion Global Initiative. I currently serve on four journal editorial boards, have served on numerous funding panels for the National Science Foundation and reviewed grant proposals on behalf of science and engineering boards in Austria, Canada, and Israel. I have served on over 124 technical program committees for selecting papers to appear at academic conferences on computer security, and I have served as program committee chair and general conference chair for multiple conferences. I have also given over 115 presentations and invited talks, including invited lectures at the University of Oxford, ETH Zurich, the Chinese Academy of Sciences, Dartmouth College, Columbia University, and UCLA. My research has been featured in news outlets such as the Wall Street Journal, BBC News, and WIRED.

10. I have experience designing and implementing communication protocols, including peripheral device communications. As a research assistant at Columbia University in 2001-2002, I incorporated authentication protocols into a Java-based software stack for the Session Initiation Protocol used for Internet

telephony. At Flarion Technologies in 2003, I implemented a packet diversion and IP header modification program in the FreeBSD operating system for mobile IP packet testing, to be deployed on WiFi devices. Since then I have led the development of mechanisms to constrain the functionality of USB devices by incorporating user expectations of device capabilities into the services that they can advertise. I also lead the development of protocols for allowing a rule-based approach to constraining malicious functionality in USB, Bluetooth, and near-field communication (NFC) devices.

11. I have experience with the development and analysis of software and firmware. In 1998 at the Royal Bank of Canada, I developed client-server applications, including design of graphical user interfaces on the client side, to port business processes from mainframe environments. Further, in 2006 I compiled and modified OpenEmbedded firmware to support the SlugOS environment to allow experimentation with storage devices, in order to develop a disk prototype for preventing a type of malware called a rootkit from persisting on a system after reboot. Since then, I have led efforts to analyze firmware on embedded devices, particularly firmware from USB flash drives that uses the Intel 8051 microcontroller, and demonstrated how such binary firmware could be lifted to an intermediate representation to enable control flow graph recovery and symbolic analysis. I also led a project that demonstrated the extraction of data from smartphone firmware

images and demonstrated that legacy command sets (i.e., “AT commands” developed for modems in the 1980s) were being used to provide undocumented features on these devices that could be accessed over a USB connection, as well as methods for enabling the transmissions of these commands.

12. I have experience with peripherals and portable devices. In course projects as an undergraduate in 1997 and 1998, I developed code for communication with a Motorola HC11 microcontroller to interface with a host computer. Since then, I have developed storage devices as discussed above that provide security services. Further, in 2009, I developed a USB device capable of attesting the integrity of the host computer it was attached to while at Symantec Research. This work was later patented (U.S. Patent 8,856,918, “Host validation mechanism for preserving integrity of portable storage data,” Petros Efstathopoulos, Bruce Montague, Dharmesh Shah, Kevin Butler, October 7, 2014). Since then, I have overseen the development of USB devices that allow for tracking the provenance of data at the block level and the development of mobile device software that automates the collection of USB enumeration data.

13. I have experience with encryption and cryptographic protocols. At Telcordia Technologies in 2000 and 2001, I provisioned and configured cable modems within a hybrid fibre-coax lab that involved secure telephony and the use of encryption over DOCSIS modems. I further led the design of developing

cryptographic primitives to support the secure delivery of routing information within the Border Gateway Protocol and have been involved with numerous projects incorporating encryption, such as mechanisms using additive homomorphic encryption to support accounting audits for preserving the confidentiality of wiretap records and cryptographic schemes for assuring the secure time release of data.

14. My curriculum vitae, included at EX1004, includes a list of publications on which I am a named author. It contains further details regarding my experience, education, publications, and other qualifications relevant to my expert opinions in this proceeding.

IV. MATERIALS CONSIDERED

15. In forming my opinions, I have reviewed the following documents and any other document cited in this declaration.

<u>Exhibit</u>	<u>Description</u>
1001	U.S. Patent No. 9,179,359 to Raleigh et al. (“the ’359 patent”)
1002	Prosecution History of U.S. Patent No. 9,179,359
1005	U.S. Pat. App. Pub. No. 2004/0224668 (“Shell”)
1006	U.S. Pat. App. Pub. No. 2008/0080458 (“Cole”)
1007	U.S. Pat. App. Pub. No. 2008/0159175 (“Flack”)
1008	International Publication No. WO 03/100581 (“Dive-Reclus”)
1009	David Flanagan, O’Reilly & Associates, Inc., “Java in a Nutshell,” 1996, ISBN: I-56592-183-6

<u>Exhibit</u>	<u>Description</u>
1010	U.S. Pat. App. Pub. No. 2009/0210537 (“Irwin”)
1011	Korhonen, “Host Identity Protocol (HIP) Implementation in the Symbian Environment,” Master of Science Thesis, Tampere University of Technology (12/10/2007) (“Korhonen”)
1012	U.S. Pat. App. Pub. No. 2006/0039354 (“Rao”)
1013	S. Lee & N. Golmie, “Power-Efficient Interface Selection Scheme using Paging of WWAN for WLAN in Heterogeneous Wireless Networks,” 2006 IEEE International Conference on Communications (Istanbul, 2006), 1742–1747 (“Lee”)
1014	U.S. Patent No. 9,215,613
1015	<i>Headwater Research LLC v. AT&T Services Inc.</i> , 2:23-CV-00397-JRG-RSP (Claim Construction Order)
1016	A.-J. Moerdijk and L. Klostermann, “Opening the networks with Parlay/OSA: standards and aspects behind the APIs.” IEEE Network, vol. 17, no. 3, pp. 58-64, May-June 2003 (“Moerdijk”)
1017	Richard Stevens et al., “UNIX Network Programming Volume 1, Third Edition: The Sockets Networking API,” 2004, ISBN: 0-13-141155-1

V. LEGAL STANDARDS

16. In forming my opinions and considering the subject matter of the ’359 patent and its claims in light of the prior art, I am relying on certain legal principles that counsel in this case explained to me. My understanding of these concepts is summarized below.

17. I understand that the claims define the invention. I also understand that an unpatentability analysis is a two-step process. First, the claims of the patent are construed to determine their meaning and scope. Second, after the claims are construed, the content of the prior art is compared to the construed claims.

18. I understand that there are two ways in which prior art may render a patent claim unpatentable. First, I have been informed that the prior art can “anticipate” a claim. Second, I have been informed that the prior art can render a claim “obvious” to a person of ordinary skill in the art. I understand that a claim is patentable if it was not anticipated and would not have been obvious by the prior art at the effective filing date of the patent.

A. Claim Construction

19. I understand that the United States Patent and Trademark Office interprets claim terms in an *inter partes* review proceeding under the same claim construction standard that is used in a United States federal court. I understand that under this standard, the meaning of claim terms is considered from the viewpoint of one of ordinary skill in the art at the time of the alleged invention.

20. I have been informed that claim terms are generally given their ordinary and customary meaning as understood by one of ordinary skill in the art in light of the specification and the prosecution history pertaining to the patent. I understand,

however, that claim terms are generally not limited by the embodiments described in the specification.

21. I understand that in addition to the claims, specification, and prosecution history, other evidence may be considered to ascertain the meaning of claim terms, including textbooks, encyclopedias, articles, and dictionaries. I have been informed that this other evidence is often less significant and less reliable than the claims, specification, and prosecution history.

B. Anticipation Under 35 U.S.C. § 102

22. I understand that a patent claim is invalid as anticipated if a single piece of prior art teaches every element of the claims, viewed from the perspective of a person of ordinary skill in the art. I also understand that an anticipatory reference does not have to recite word for word what is in the anticipated claims. Anticipation can also occur when a claimed limitation is inherent in the relevant reference. I have been advised that if the prior art necessarily functions in accordance with, or includes, the claims limitations, it can anticipate even though the limitation is not expressly disclosed.

C. Obviousness Under 35 U.S.C. § 103

23. I understand that a patent claim is invalid as obvious if the claimed invention would have been obvious to a person of ordinary skill in the art at the time the claimed invention was made. This means that even if all of the elements of the

claim cannot be found in a single prior art reference that would anticipate the claim, a person of ordinary skill in the field who knew about all the prior art would have come up with the claimed invention. I understand that in an obviousness determination, the person of ordinary skill in the art is presumed to have knowledge of all prior art. I understand that whether a claim is obvious is based upon the determination of several factual issues.

24. I have been informed that, in an obviousness analysis, prior art must be analogous to the patent being considered. I have been informed that a prior art reference is considered to be analogous, or in the same field of art, if the reference is either (1) in the same field of endeavor as the challenged patent, regardless of the problems the challenged patent and the prior art address, or (2) reasonably pertinent to the particular problem being solved by the challenged patent.

25. In considering obviousness, I understand that one must determine the scope and content of the prior art. I also understand that one must determine what differences, if any, existed between the claimed invention and the prior art.

26. I understand that a patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art. In evaluating whether such a claim would have been obvious, one may consider whether a reason has been identified that would have prompted a person of ordinary skill in the art to combine the elements or concepts from the prior

art in the same way as in the claimed invention. There is no single way to define the line between true inventiveness on the one hand (which is patentable) and the application of common sense and ordinary skill to solve a problem on the other hand (which is not patentable). For example, market forces or other design incentives may be what produced a change, rather than true inventiveness.

27. I also understand that, while there is no requirement that the prior art contain an express suggestion to combine known elements to achieve the claimed invention, and while a suggestion to combine known elements to achieve the claimed invention may come from the prior art as a whole or individually and may consider the inferences and creative steps a person of ordinary skill in the art would employ, as filtered through the knowledge of one skilled in the art, obviousness grounds cannot be sustained by mere conclusory statements and must include some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. Moreover, I understand that a proper obviousness analysis requires a determination of whether a person of ordinary skill in the art would have a “reasonable expectation of success”—but not “absolute predictability” of success—in achieving the claimed invention by combining prior art references.

28. I understand that one may consider whether (1) the change was merely the predictable result of using prior art elements according to their known functions, or whether it was the result of true inventiveness; (2) there is some teaching or

suggestion in the prior art to make the modification or combination of elements claimed in the patent; (3) the innovation applies a known technique that had been used to improve a similar device or method in a similar way; (4) the claimed invention would have been obvious to try, meaning that the claimed innovation was one of a relatively small number of possible approaches to the problem with a reasonable expectation of success by those skilled in the art; (5) the invention merely substituted one known element for another known element in order to obtain predictable results; (6) the invention merely applies a known technique to a known device, method, or product to yield predictable results, or (7) known work in the field may have prompted variations of use of the same inventions in the same or different fields due to market forces or design incentives that would have been predictable to a person of ordinary skill in the art.

29. I understand that one must be careful not to determine obviousness using the benefit of hindsight; many true inventions might seem obvious after the fact. I understand that the determination should be based on the position of a person of ordinary skill in the field at the time the claimed invention was made and should not consider what is known today or what is learned from the teaching of the patent.

D. Secondary Considerations of Non-Obviousness

30. I understand that certain secondary considerations may be considered in evaluating obviousness in order to prevent hindsight bias. These secondary

considerations include commercial success of products that practice the patent, long-felt need for the patented technology, failure by others to solve the problem addressed by the patent, initial skepticism by others in the industry, industry recognition and praise of the patented products, and efforts by others to copy the patented technology. I am not presently aware of any secondary considerations that would apply here.

31. I have been informed that these secondary considerations of non-obviousness must be balanced against the strength of the prima facie case of obviousness. Where the invention represents no more than the predictable use of prior art elements according to their established functions, secondary considerations are inadequate to establish non-obviousness.

32. I have also been informed that the patentee must establish a nexus between any secondary consideration factors and the claimed invention. Where a secondary consideration factor results from something other than what is both claimed and novel in the patent, there is no nexus to the merits of the claimed invention.

33. I have been informed that, for example, the nexus requirement as it applies to the evidence of commercial success requires that the patentee must show that the sales of a product that allegedly practices the claimed invention must be a direct result of the unique characteristics of the claimed invention, as opposed to

other economic and commercial factors that are unrelated to the patented invention. I understand that this means if the commercial success is due to an element that existed in the prior art or that is not claimed by the patent, then no nexus exists.

34. I have been further informed that, for example, the nexus requirement as it applies to the long-felt need element must likewise be supported by evidence. Where the differences between the prior art and the claimed invention are minimal, it cannot be said that any long-felt need was met by the patented invention or that the patent solved any technological problems that were unaddressed by others.

35. I have been informed that evidence of industry praise must also show a nexus with the claimed invention, and that self-serving statements by the patentee regarding the merits of the claimed invention are generally not given weight.

VI. TECHNOLOGY OVERVIEW

36. In this section I provide a broad overview of the relevant technology and state of the technical community at the asserted priority date (or Critical Date) of the '359 patent. This description is not intended to be exhaustive, but to provide sufficient background to support and contextualize my analysis below.

A. Mobile Devices

37. In 2009, use of smartphones experienced a significant surge, marking a pivotal moment in the evolution of mobile technology. The year saw a growing number of consumers transitioning from traditional feature phones (e.g., flip phones)

to more capable and versatile smartphones. Key factors driving this uptake included advancements in mobile operating systems such as Apple's iOS and Google's Android, offering a user-friendly interface and access to a burgeoning app ecosystem. Additionally, the earlier introduction of 3G networks enabled faster mobile data speeds, prompting users to embrace smartphones for tasks beyond voice and messaging. As a result, smartphone sales soared, laying the groundwork for the transformative impact they would have on communication, productivity, and entertainment in the years to come.

38. However, smartphones in 2009 were resource-limited compared to the current devices. For example, smartphones of 2009 were equipped with single-core processors, usually running at speeds below 1 GHz. These processors lacked the computing power of today's multi-core CPUs found in modern smartphones. As a result, tasks that required significant processing, such as running complex applications or playing graphically intensive games, were often slow and less efficient. Similarly, (non-transitory) memory capacities in smartphones of 2009 were generally limited, typically ranging from 128 MB to 512 MB. Long term storage was also limited, consisting of between 4 and 16GB of space. Devices were

connected to wireless networks using WWAN (cellular) and WLAN (Wi-Fi) modems.¹

39. The networks that were being used in 2009 were also limited in speed and throughput. In 2009 wide-area (WWAN) cellular 3G networks, as implemented by the Global System for Mobile Communications (GSM)² and Code Division Multiple Access (CDMA) protocols were becoming widespread, but 4G networks were still in the early stages of deployment. This meant slower Internet speeds and limited coverage for mobile data, affecting the overall online experience. Wi-Fi technology as implemented in the 802.11 suite of protocols complemented cellular networks. Smartphones, laptops, and other devices could connect to Wi-Fi hotspots in homes, offices, cafes, and public spaces to access the Internet at much higher speeds than what 3G networks could offer. Wi-Fi also allowed users to offload data

¹ The term “modem” refers to a modulator/demodulator, which is a device to process electrical signals received and sent over a physical media (e.g., copper wire) or over the air. In mobile devices, modems are software-supported radios that process specific network access protocols.

² Note that GSM predates 3G networks (actually back to 1991), but, for the purposes of this report, I will focus only on those variants of GSM which were classified as 3G.

usage from their cellular plans, which was particularly helpful due to the relatively limited data caps on mobile plans during that time. Regardless of the network protocol it was running on, the IP protocol was used to connect each device with the Internet.

40. At the same time, battery technology had not yet seen significant advancements, leading to smartphones with relatively short battery life. The computational requirements and increased user engagement (using the phone for longer periods) led to increased demand for energy. Here, the combination of power-hungry processors, smaller batteries, and less optimized software meant that users often had to charge their phones frequently.

41. While 2009 smartphone displays were already evolving towards touchscreens, they were smaller, less vibrant, and had lower resolutions compared to modern displays. Common screen resolutions were around 320x480 pixels, making it challenging to display content with high visual fidelity. Device operating systems provided windowed environments that would generally have a single application being displayed (and prioritized by the operating system) and interacting with the users. These applications were referred to as having focus (of the user interface). Users could switch between applications as needed but could only interact with the one application with focus at a time.

B. Resource Management in Mobile Devices

42. A key challenge in 2009 was finding ways to address the rapidly increasing needs of mobile users using the resource-limited devices and networks. Solutions to these problems were developed by adapting known techniques in computer systems and networks (e.g., priority queuing), as well as developing new, faster devices and networks.

43. Policy for computational systems has been a part of the technical landscape since the 1970s, with a primary focus on differentiating resource usage in various environments. For example, a policy might specify that data from interactive applications, such as multi-player games, are assigned the highest priority to ensure the best user experience. These policies were implemented to manage and allocate computing resources efficiently in diverse computing environments. For instance, differentiated services emerged as a crucial policy in this context, especially as mobile devices such as phones began to traverse heterogeneous environments, transitioning between Wi-Fi and cellular networks seamlessly. By employing policies like differentiated services, computing systems could optimize resource allocation, ensuring a smooth and reliable user experience while adapting to changing network conditions and device capabilities in these dynamic environments. Such techniques were particularly important to address the challenges faced by early mobile devices.

44. A Policy Decision Point (PDP) and Policy Enforcement Point (PEP) are essential components in policy-based systems. The PDP functions as the determining logic of the system, responsible for evaluating policy rules and making decisions based on the input received from sources. Conversely, the PEP serves to enforce policy. PEPs are located at the point where policies are applied. The PEP communicates with the PDP to obtain the relevant policy decisions and then enacts them by controlling access to resources. In the example given above, the PDP would determine whether data are from an interactive game, and if so, the PEP would assign the highest priority to the data, e.g., by adding the data to a high- priority queue. Again, this would ensure the user does not experience performance problems (e.g., lag).

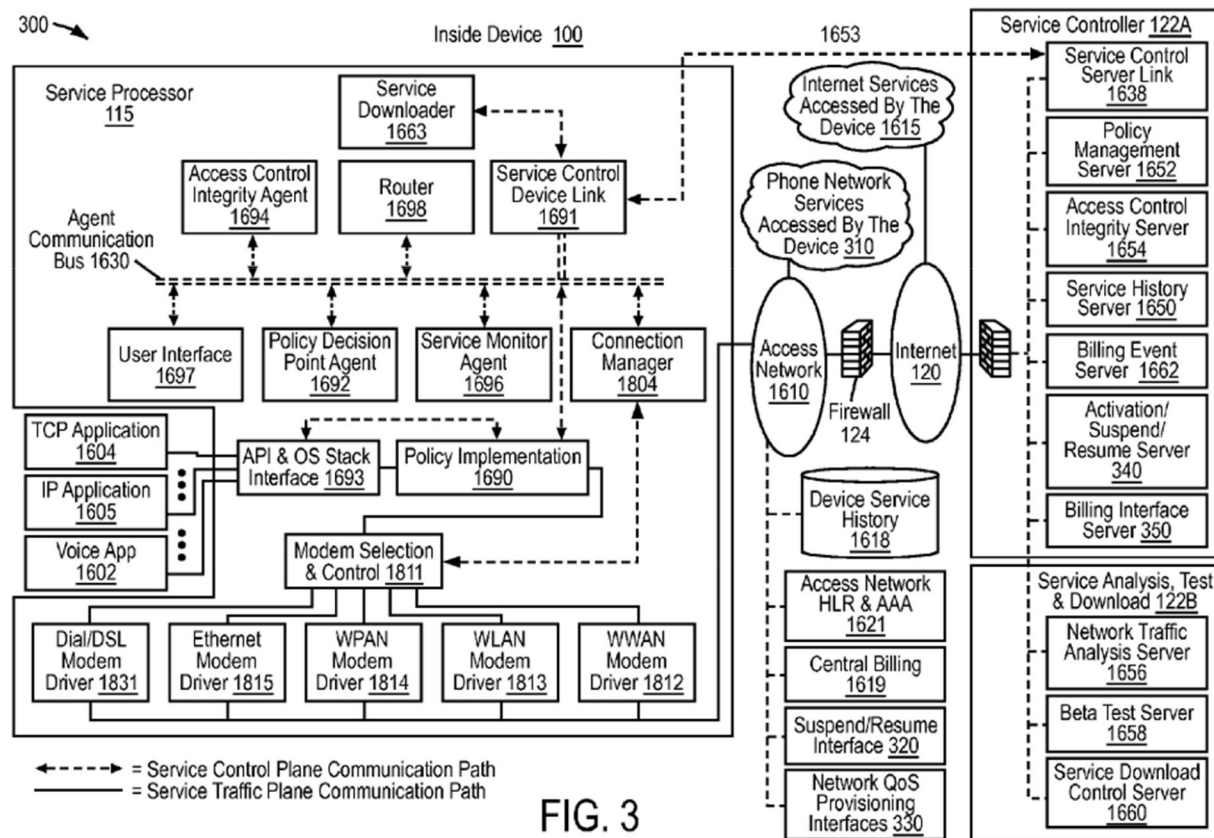
VII. THE '359 PATENT

A. Specification

45. The '359 patent is directed to a “wireless end-user device with differentiated network access status for different device applications.” EX1001, Title. The '359 patent explains that “a wireless end-user device has a wireless wide-area network (WWAN) modem and supports multiple wireless network types,” which can include a WLAN modem, as illustrated below. EX1001, Abstract, FIG. 3. The '359 patent teaches that “in wireless networks, managing the wireless access connection capacity and network access connection resources is important to

maintain network performance as network resources/capacity demand increases.”

EX1001, 3:44-4:5.



EX1001, Fig. 3.

46. To manage capacity, the device can “apply a service usage control policy for the network service usage activities to protect network capacity (e.g., reduce network capacity demand).” EX1001, 11:16-12:2. The device can “assign[] a priority to a network service usage activity and control[] traffic associated with the network services usage activity based on the assigned priority” and can provide a “differentiated and dynamic background services classification ... as a function of network availability state and/or network busy state.” EX1001, 12:3-10. The service

usage control policy can assist “in network access control of network service usage activities (e.g., deferring some or all of the network capacity demand from these source activities).” EX1001, 12:11-27. The device can associate “the network service usage activity with a network service usage control policy” based on factors such as “a network busy state, a time of day, ..., an application behavior, ..., a power state of device, a device usage state, a time based criteria, availability of alternative networks, and a set of policy rules.” EX1001, 85:9-20. When a policy applies to an access request, the device can take various actions including acting to “throttle, delay, priority queue, time window, suspend, quarantine, kill, remove, and other well known traffic control techniques.” EX1001, 13:10-30.

47. To enforce the controls, the device includes “a policy implementation driver/agent 1690 (e.g., policy enforcement point driver/agent)” and a Policy Decision Point Agent 1692. EX1001, Fig. 3, 57:34-58:17. A person of ordinary skill in the art at the time of the earliest claimed filing date of the '359 patent (“POSITA”) would have recognized policy decision points (PDPs) and policy enforcement points (PEPs) as terms of the arts describing components that determine whether a policy applies (PDP), and when the policy applies, enforcing the policy decision (PEP). EX1010, ¶20.

B. Prosecution History

48. Application Serial No. 14/673,625 was filed with 1 original claim, which was replaced in a preliminary amendment. EX1002, 691-98, 873. The Examiner allowed these claims without issuing any Office Action. EX1002, 122-28. Shell and Flack were not before the Examiner, and the Examiner never applied Cole (or references related to Cole) or advanced any substantive rejection based on these references. EX1001, Cover.

VIII. CLAIM CONSTRUCTION

49. I understand that, in this proceeding, the claims are construed according to their ordinary and customary meaning, in light of the specification and prosecution history, as understood by a person of ordinary skill in the art at the time of the invention. I understand that this is the same claim construction approach used in district court litigation.

50. I have been instructed that the “intrinsic record” includes the ’359 patent itself, including the claims, description, and figures (EX1001), and the ’359 patent’s prosecution history—i.e., the record of proceedings at the U.S. Patent and Trademark Office (“USPTO”) concerning the patent (EX1002). I understand that, like the claims and written description, the prosecution history provides evidence to a person of ordinary skill in the art of how the inventor intended his patent to be understood, and how the USPTO understood the patent. I understand that the

inventor is permitted to apply a special definition to the terms or to limit the scope of claim terms in his patent claims, which may differ from the term's plain and ordinary meaning. That special definition or limitation on scope may be provided in the patent's written description, the patent's prosecution history, or both.

51. I understand that claim interpretation may also be informed by “extrinsic evidence” (that is, evidence outside of the patent record itself). I have been informed that extrinsic evidence may include dictionaries, technical treatises, and other materials evidencing the meaning of a claim term and the understanding held by a person of ordinary skill in the art in the relevant time period.

In my opinion, none of the claim terms in the '359 patent require an express construction for the purpose of review by the Patent Office, and should all be given their plain and ordinary meaning as they would have been understood by a person of ordinary skill in the art at the earliest priority date of the '359 patent. I have been asked to also consider how that the term “differential traffic control policy” is found in at least one other Headwater patent (U.S. Patent No. 9,215,613, EX1014) that claims priority to some of the same applications as the '359 patent. EX1014, cover. I was informed that the '613 patent was asserted in *Headwater Research LLC v. AT&T Services Inc.*, 2:23-CV-00397-JRG-RSP and the parties presented constructions related to the term “differential traffic control policy.” EX1015, 21-22. I understand that the court construed the term to mean a “rule for controlling

network traffic that distinguishes between two or more things.” EX1015, 21-22. I discuss this construction below.

IX. LEVEL OF ORDINARY SKILL

52. I am informed that patentability must be analyzed from the perspective of “one of ordinary skill in the art” in the same field as the patent-in-suit at the time of the invention. I am also informed that several factors are considered in assessing the level of ordinary skill in the art, including (1) the types of problems encountered in the art; (2) the prior art solutions to those problems; (3) the rapidity with which innovations are made; (4) the sophistication of the technology; and (5) the education level of active workers in the field. I have been asked to assume that the priority date (or Critical Date) of the ’359 patent is January 28, 2009.

53. In my opinion, a person of ordinary skill in the art pertinent to the ’359 patent as of its earliest priority date would have at least a bachelor’s degree in computer science, electrical engineering, or a similar field, and approximately two years of industry or academic experience in a field related to computer software development and/or computer networking. More related education would compensate for fewer years of work experience (and vice versa). I meet this definition of one of ordinary skill in the art now and at the priority date of the ’359 patent.

X. OVERVIEW OF PRIOR ART

A. Shell

54. Shell focuses on “[i]ncreasing the level of automation when establishing and managing network connections.” EX1005, Abstract. This is accomplished using “connection manager” modules that “relieve[] users from having to know detailed information about [a] remote resource and [a] destination network.” EX1005, Abstract. Specifically, “[w]hen the connection manager receives a request to access a remote resource,” it uses “extensible decision logic to identify a most appropriate connection method for connecting to a destination network.” EX1005, Abstract. Through its connection manager modules, the Shell system “abstract[s] connection information from a user so as to limit required user interactions when creating network connections.” EX1005, ¶3.

55. Shell recognizes that “conditions associated with a network may make establishment of a connection using a particular connection method undesirable at certain times,” e.g., if a network is experiencing “heavy usage” or “high degree of latency.” EX1005, ¶7. This is because a user may not be “aware of the costs associated with different types of connections” and thus may use a “less desirable connection method to access a resource.” EX1005, ¶7. Beyond this issue, Shell attributes “[o]ther problems associated with connection methods [that] also exist at the programming level.” EX1005, ¶9. Because “[m]ost connection methods utilize

different Application Program Interfaces ('APIs') when establishing a connection[,]" they may require different APIs or "different parameters," which "[r]equires programmers to know the nuances of different connection methods" and thereby may be "time consuming" and lead to reduced efficiency. EX1005, ¶9. Shell thus focuses on "automating the establishment and management of various kinds of network connections in a manner that reduces the amount of information a user must know to establish a connection." EX1005, ¶11. In this way, "[a] user does not need to identify the underlying properties of the network where the resource is located or a connection method for getting to that network." EX1005, ¶12.

56. During operation, "an application included in a computing device may request access to a remote resource located in a destination network." EX1005, ¶33. A connection manager (or "connection planner") "analyzes a variety of factors to determine the most appropriate connection method to connect to the destination network." EX1005, ¶33. Examples of factors include "local resources currently available on the computing device, the priority of the request, or connection parameters (such as latency, cost, and bandwidth) associated with a connection method." EX1005, ¶33. "When the most appropriate connection method is known, a connection service provider associated with the connection manager may cause a system level [API] to create a connection from a local computing device to the destination network." EX1005, ¶33.

57. Figure 1 shows an example of a “suitable computing environment” for the techniques disclosed throughout Shell. EX1005, ¶37.

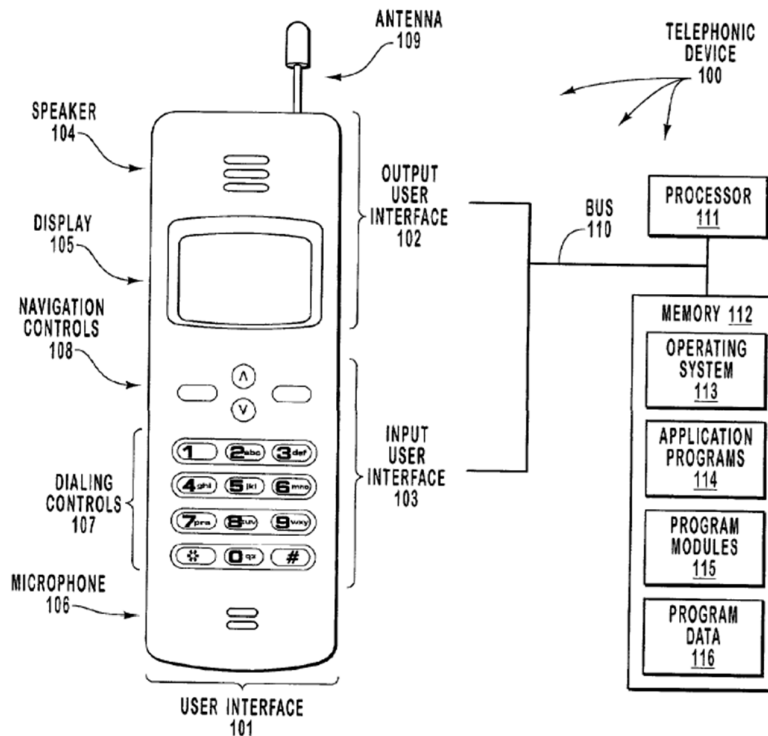


FIG. 1

EX1005, Fig. 1.

58. The operating environment includes a “telephonic device 100” as a “general-purpose computing device” with standard computing elements for “complex and flexible general-purpose processing capabilities.” EX1005, ¶¶39-44.

59. Figure 2 shows an “arrangement” of “functional components” that “offers a level of abstraction between application modules that may request a connection, and the system level APIs that actually establish the connections.” EX1005, ¶45.

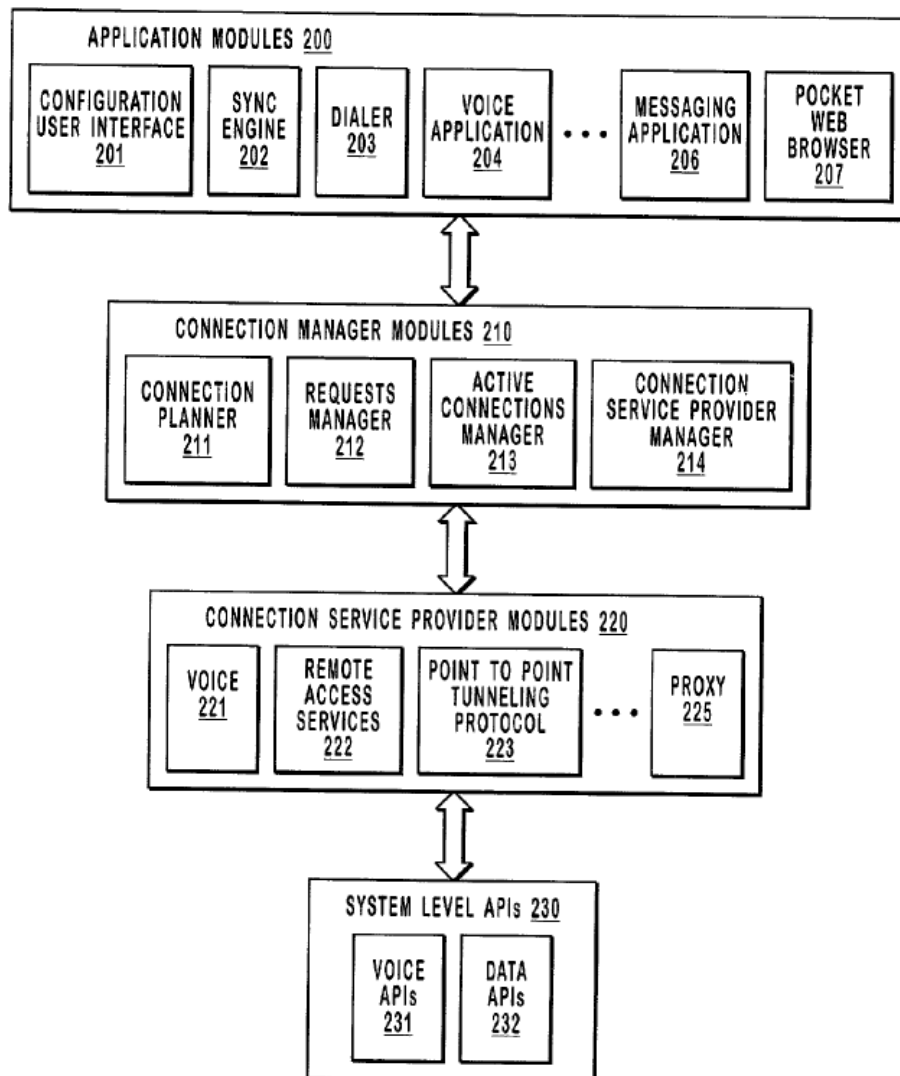


FIG. 2

EX1005, Fig. 2.

60. Specifically, “[c]onnection manager modules 210 and connection service provider modules 220 may be configured as APIs that receive standard data when being called by another module.” EX1005, ¶46. These modules “include computer-executable instructions that cause, directly or indirectly, the appropriate

calls to be sent to system level APIs 230[,]” thereby simplifying the process of establishing connections for application modules 200. EX1005, ¶46. Further, “APIs associated with connection manager modules 210 and connection service provider modules 220 may be integrated into protocol stacks.” EX1005, ¶48. Thus, “APIs associated with connection manager modules 210 and connection service provider modules 220 may be called instead of APIs associated with conventional protocol stacks.” EX1005, ¶48.

61. Further, “connection manager modules 210 may cause resources to be released from and allocated to connection methods.” EX1005, ¶92. “When this occurs, connection manager modules 210 may notify application modules 200 of such changes[,]” e.g., “application modules may be notified that a connection will be established, that a connection is currently active, that connection is being disconnected, that no connection is currently active, that a connection to a resource is blocked for some reason....” EX1005, ¶92. Connection manager modules 210 may also “provide connection specific parameters to application modules 200 including, bandwidth, latency, power consumption, cost, physical location, past connection successes, or other parameters....” EX1005, ¶92.

62. Figure 5A shows an example of a technique of network connection request handling based on prioritization of network connection requests. EX1005, ¶57.

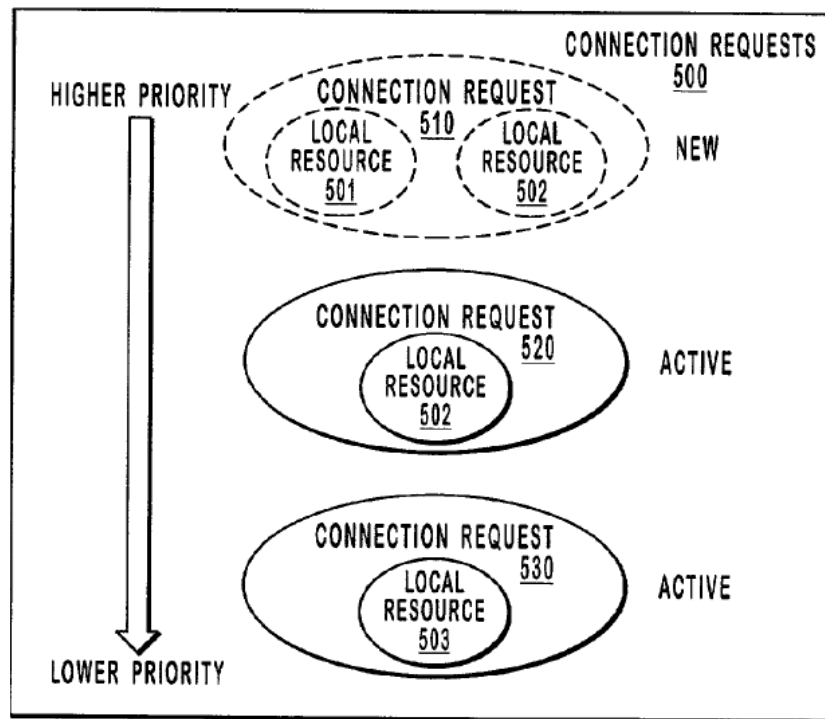


FIG. 5A

EX1005, Fig. 5A. In the example above, “connection requests 500 includ[e] new connection request 510 and active connection requests 520 and 530.” EX1005,

¶57. “Connection requests 510, 520, and 530 are prioritized, with connection 510 having the highest priority and connection 530 having the lowest priority.”

EX1005, ¶57. In this example, “requests from sync engine 202 may be assigned a higher priority than requests from messaging application 206[,]” though

“[p]riorities may also be assigned to connection requests based on the current state of applications associated with the connection requests[,]” e.g., “a request from an application that is currently active may be assigned a higher priority than a request from an application running in the background.” EX1005, ¶57. Other examples of

prioritization are “based on characteristics associated with the connection requests[,]”: e.g., “voice requests may be assigned a higher priority than data requests.” EX1005, ¶57.

63. As depicted in Figure 5B, Shell’s connection manager allocates resources (e.g., provides an application with access to a network) based on the priority associated with a connection request. EX1005, ¶¶60-66, Fig. 5B.

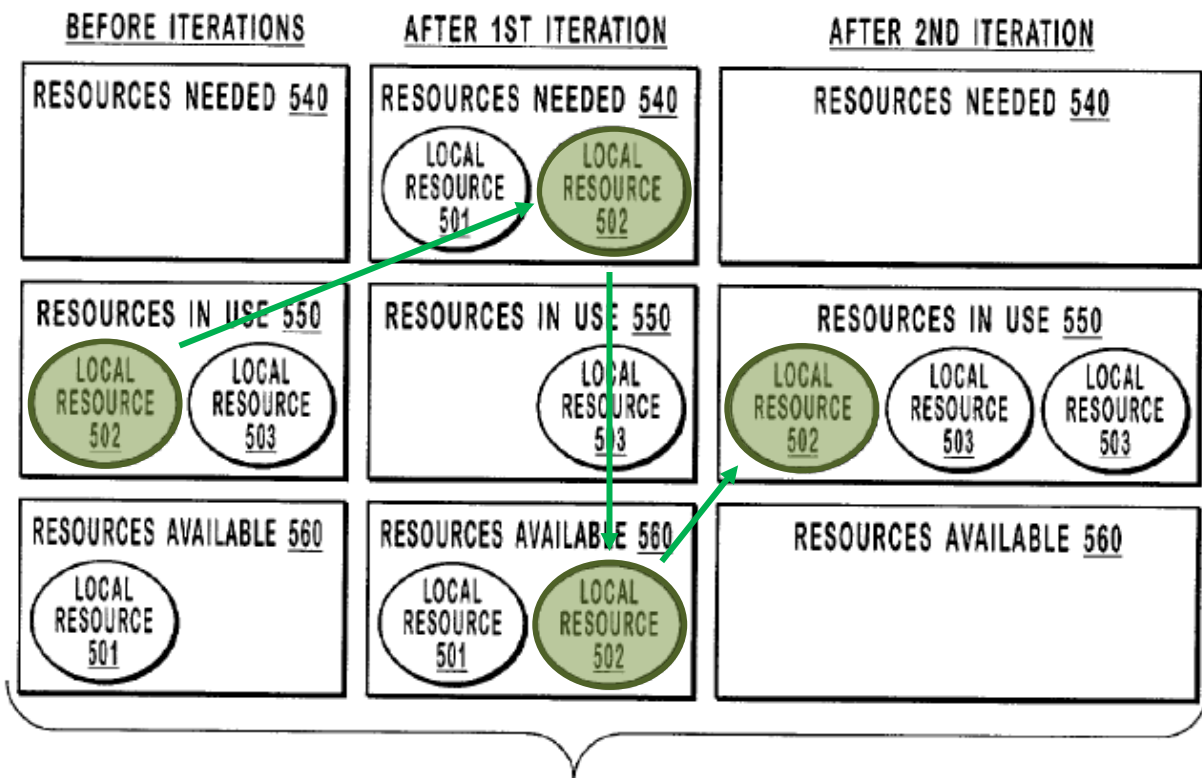


FIG. 5B

EX1005, Fig. 5B (annotated). For example, at the start of the process (“Before Iterations”), resources 502 and 503 are in use (by connection requests 520 and 530), and resource 501 is available. EX1005, ¶60. When a new connection request

510 is made (first iteration), the connection manager registers that new connection request 510 needs local resources 501 and 502. EX1005, ¶¶61-66. As the process continues, the connection manager weighs the priority of the new request 510 against the active connections 520 and 530 to determine how to allocate the resources. EX1005, ¶67. Because new request 510 has a higher priority than request 520, the connection manager disconnects the application associated with request 520 to resource 502. EX1005, ¶67. The connection manager then allocates resources 501 and 502 to new request 510, which provides the application associated with new request 510 a connection. EX1005, ¶67. The above process describes that resource 502 is made unavailable to connection request 520 but is available to new request 510. EX1005, ¶67.

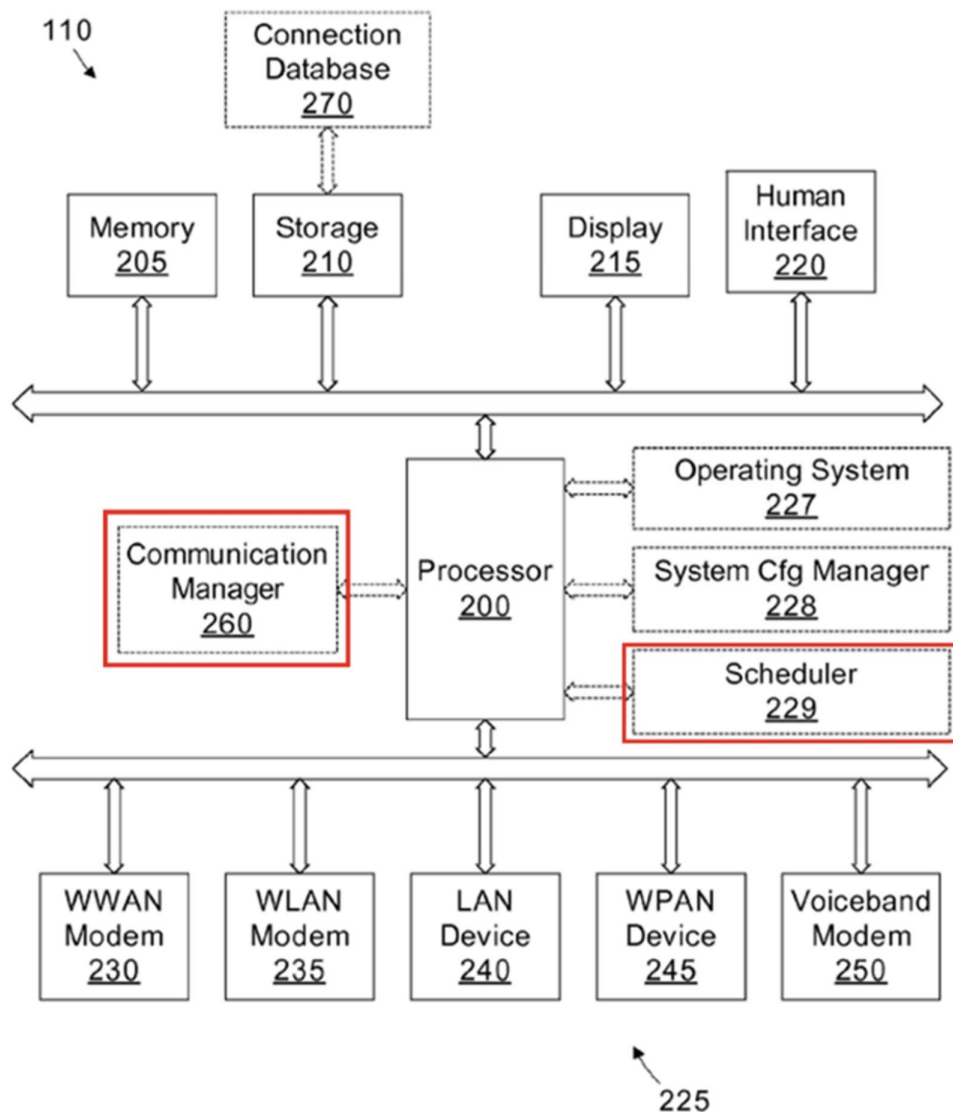
64. Shell further teaches embodiments in which “an application may notify connection [planner] 211 it wishes to ‘piggy back’ on other connection requests,” i.e., “an application may request to only access resources in a destination network when another application is already accessing resources.” EX1005, ¶70. In such embodiments, “[a]pplications that wish to piggy-back, may call APIs associated with connection manager modules 210 and be given an open handle.” EX1005, ¶70. Open handles are not always given as the API call may not be successfully fulfilled, for example, if a required resource is not available. The open handle “facilitates

notification back to the application that a connection on which the application may want to piggy back has been established.” EX1005, ¶70.

65. Shell is analogous art to the ’359 patent. For example, Shell is in the same field of endeavor: management of an application’s ability to connect to wireless networks. *Compare*, EX1005, Abstract, ¶¶57-67, *with* EX1001, Abstract. Shell is also reasonably pertinent to at least one problem addressed by the ’359 patent: optimizing use of data on an application-by-application basis. *Compare*, EX1005, ¶¶69-70, 12, *with* EX1001, 4:65-5:20.

B. Cole

66. Cole describes “[a] method for configuring a mobile device including a plurality of communication interfaces.” EX1006, Abstract, ¶14. A “connection manager 260” selects “communication interfaces” for connecting to remote networks (e.g., a wireless local area network (WLAN), a wireless wide area network (WWAN), and a wireless personal area network (WPAN)). EX1006, ¶¶15, 3, 31, 35, 36, Figs. 1, 2. The connection manager can also “schedule various events” using a “scheduler” (also annotated) to “effectively use available bandwidth.” EX1006, ¶¶74, 36, 118, 119. This includes taking action when “monitored loads” exceed a “predetermined threshold.” EX1006, ¶¶75, 77.



EX1006, Fig. 2 (annotated).

67. The connection manager manages communication interfaces to “determine which connections should be active at any given time.” EX1006, ¶36. This determination includes monitoring various network connectivity metrics, such as battery power. EX1006, ¶¶36, 69, 80.

68. To assist in evaluating communication interfaces, the connection manager “maintains data related to each of the communication interfaces,” including “connection name” and “connection technology type.” EX1006, ¶37. This information may be “based on historical data[,] ... user specified parameters, or static parameters,” and may also be “monitored” by the connection manager. EX1006, ¶38.

69. When “the monitored load exceeds a predetermined threshold (e.g., 85% of capacity), the connection manager 260 may take one of several actions to manage the load.” EX1006, ¶75. Applications can be “aborted,” “postpone[d],” cancelled and rescheduled, and “block[ed].” EX1006, ¶¶80, 85-90, 125. These actions can also be taken in response to mobile device states, including a “power state.” EX1006, ¶¶90-98. “By monitoring the load on an active connection, the connection manager 260 tries to proactively prevent increased latency from an overloaded connection from impacting the performance of the running applications or the efficiency of the user.” EX1006, ¶75.

70. Cole is analogous art to the ’359 patent. For example, Cole is in the same field of endeavor: management of an application’s ability to connect to wireless networks. *Compare*, EX1006, ¶¶68, 73, 80, *with* EX1001, Abstract. Cole is also reasonably pertinent to at least one problem addressed by the ’359 patent: optimizing use of data on an application-by-application basis. *Compare*, EX1006,

¶¶73-74, 80, 12, *with* EX1001, 4:65-5:20. As addressed further below, a POSITA would have clearly observed several obvious and beneficial reasons to combine features of Cole with Shell.

C. Flack

71. Flack discloses systems and methods of “selectively establish[ing] network connectivity by monitoring, on a client, communications activity incident to accessing a network.” EX1007, Abstract. Flack’s client-side system may be executed on a wireless phone or other computing device. EX1007, ¶¶38, 62. Once communication activities are detected, the system “identif[ies] an application associated with the communications activity,” accesses a “communications configuration associated with the application,” and “selectively establish[es] network connectivity based on the communications configuration.” EX1007, ¶¶3, 17, 50, 54.

72. Flack identifies and seeks to address inefficiencies in maintaining constant network connectivity. For example, Flack recognizes that “background and ambient processes, such as spyware protection and update tools generate small amounts of traffic.” EX1007, ¶16. Processes like these unnecessarily keep network connections active even when they “do not represent mission critical communication or represent communications that may be exchanged at a later time when communications do not incur the same costs for the user.” EX1007, ¶16. Flack

teaches that “[i]t might be better to establish and preserve connectivity for only selective user applications,” providing the example of only providing connectivity for applications that involve “user actions that generate visual feedback useful to the user in perceiving the user application.” EX1007, ¶16.

73. Flack discloses a system that uses a configurable profile to determine how and when applications may access a network. This includes applying policies for various applications such as email, instant messaging, web browsing, and media playback. EX1007, ¶¶22, 31, 50, 55, 59-61. The system is configured to distinguish between user-initiated applications and non-user-initiated applications. EX1007, ¶¶8, 50. Considering user activity in its policy determination allows the system to allow applications to connect to a particular network only when the user is using an application (EX1007, ¶19) or to allow a background connection only if the network connection is already established by a user application (EX1007, ¶24). *See also* EX1007, ¶¶5-6, 42, 48-58, Figs. 4, 4A.

74. Flack is also analogous art to the ’359 patent. For example, Flack is in the same field of endeavor: management of an application’s ability to connect to wireless networks. *Compare*, EX1007, ¶¶16, 17, *with* EX1001, Abstract. Flack is also reasonably pertinent to at least one problem addressed by the ’359 patent: optimizing use of data on an application-by-application basis. *Compare*, EX1007, ¶17, *with* EX1001, 4:65-5:20. As addressed further below, a POSITA would have

immediately appreciated strong motivations to combine features of Flack with Shell-Cole.

XI. GROUND 1: SHELL AND COLE RENDER OBVIOUS CLAIMS 1 AND 26

75. In my opinion, Shell and Cole render claims 1 and 26 obvious because the references' combined teachings disclose every limitation found in these claims.

A. Motivations to Combine

76. In my opinion, while Shell's prioritization techniques efficiently utilized network resources, in mobile devices, power management is also critical, at least because applications cannot execute without sufficient power. Network transmissions from a mobile device can cause high power utilization, especially using some types of power-intensive network connections (e.g., WWAN). EX1006, ¶11. Thus, when a high rate of network connection requests are provided by applications, power demand can increase, highlighting the importance of power management. EX1006, ¶¶11, 78. Indeed, Shell recognizes the need for power management in mobile devices, explaining that factors used to identify most appropriate connection techniques include "power consumption" and "current battery status," among others. EX1005, ¶16. Shell explains that its "[c]onnection manager modules 210 may also provide connection specific parameters to application modules 200 including ... power consumption, ... or other parameters

previously discussed. EX1005, ¶92. A POSITA reading Shell would thus have clearly been taught and motivated to consider power consumption and battery status as part of Shell's decisions regarding what connection techniques to use for individual applications and individual communication requests. Indeed, as I discussed above, power management and efficiency were known and important goals for POSITAs. *See* Section VI.

77. By the Critical Date, one well-known technique for mobile device power management was selectively adjusting between available network connections (e.g., WWAN, WLAN). *See* EX1013, 1. In addition to reducing power consumption, this technique could also increase available bandwidth, reducing network congestion. EX1006, ¶78. WLANs typically provide more bandwidth and involve lower cost relative to that involved with use of WWANs (and further involve less power usage). This would have motivated a POSITA to seek solutions beyond Shell.

78. A POSITA would have looked to references directed to controlling network connectivity for mobile devices to address related concerns, e.g., cost, power, network selection. EX1006, Abstract, ¶¶14-15. Cole offers Internet-access control techniques that address excess power consumption and selection of an appropriate network. EX1006, ¶¶78, 83, 88, 98. In various examples, Cole determines whether to apply an Internet access control (connecting using the "best"

connection technique) based on a condition (power state change of the device and high load on the current communication link from device usage). EX1006, ¶¶83, 98. Integration of these techniques into Shell's device would have therefore allowed power savings when determining network-access connectivity.

79. In operation, the Shell-Cole device first evaluates network and power conditions using Cole's techniques to determine whether to apply an Internet-access control to connect to the network with the "best" connection type, then evaluates further conditions (e.g., whether an application is running in the foreground or background, which application is running, etc.) using Shell's techniques to determine the priority of the packets traveling over the network selected by the Shell-Cole device using Cole's techniques. EX1005, ¶92; EX1006, ¶¶11, 78, 83, 88, 98.

80. A POSITA would have incorporated functionality of Cole's communication manager (communication manager 260) into Shell's connection manager modules (connection manager modules 210). The Shell-Cole device would have included a network interface containing WWAN Modem 230 (supporting 2G and 3G cellular technologies) and WLAN Modem 235 (supporting Wi-Fi) from Cole. EX1005, Fig. 1; EX1005, ¶31. A POSITA would have been motivated to do so because, using such connections, the Shell-Cole device would establish and manage "various kinds of network connections." EX1005, ¶32. In other words, it would have improved flexibility in establishing connections. Indeed, to the extent

not explicitly disclosed in Shell, the modification to Shell achieves the ability to connect to these wide area networks, allowing the Shell-Cole device to send and receive data packets required for Internet services like web browsing, email, and media content. A POSITA would have further been motivated to incorporate functionality of Cole's communication manager (communication manager 260) into Shell's connection manager modules (connection manager modules 210) to ensure the mobile device was efficient from a power and battery usage standpoint. While Shell teaches selecting connection techniques based on "power consumption" and "current battery status," EX1005, ¶16, Cole provides more implementation details and optimizations to achieve power and battery efficiency. *See, e.g.*, EX1006, ¶87 ("determining if sufficient batter[y] power is present to complete the transaction"), ¶¶89-90 (connection manager 260 changing configurations "in direct response to the power state of the mobile device 110"), 92 (considering "power and thermal characteristics"), ¶¶91-94 (considering whether "AC power" is available). These would have been compelling motivations from the standpoint of a POSITA because, as I discussed above, power and battery efficiency were foremost goals in the field of mobile devices before the Critical Date. *See* Section VI.

81. The Shell-Cole device would have otherwise remained unchanged, including elements found in Shell's mobile device, such as a processor (e.g., Shell's CPU 111), non-transitory memory (e.g., Shell's memory 112), and a display (e.g.,

Shell's display 105). EX1005, Fig. 1, ¶¶38-44. These elements are similarly found in Cole. EX1006, Fig. 2, ¶34. The processor of the Shell-Cole device would have been configured to perform the operations discussed in Ground 1 because it "may allow for complex and flexible general-purpose processing capabilities." EX1006, ¶40.

82. A POSITA would have had a reasonable expectation of success in incorporating power management functionality (as taught by Shell and explained in further detail by Cole) into the connection manager modules taught by Shell. Such a combination would have required only routine programming knowledge well within the skill of a POSITA before the Critical Date. Indeed, the change would have amounted to nothing more than the use of a known technique to improve similar devices—in each instance, a mobile device equipped with network interfaces—in a similar way and combining prior art elements according to known methods to yield the predictable results described above. EX1005, ¶¶36, 40; EX1006, Fig. 2, ¶¶34-35. A POSITA would have known that processors commonly run processes such as those run by the software modules described in Shell and Cole. Therefore, the combination of the techniques of Cole into Shell in the Shell-Cole device would require only routine programming knowledge.

B. Independent Claim 1

1. [1pre]: “A *wireless end-user device*, comprising:”

83. In my opinion, Shell-Cole renders [1pre] obvious. Shell’s Figure 1 depicts a telephonic device 100 (*wireless end-user device*):

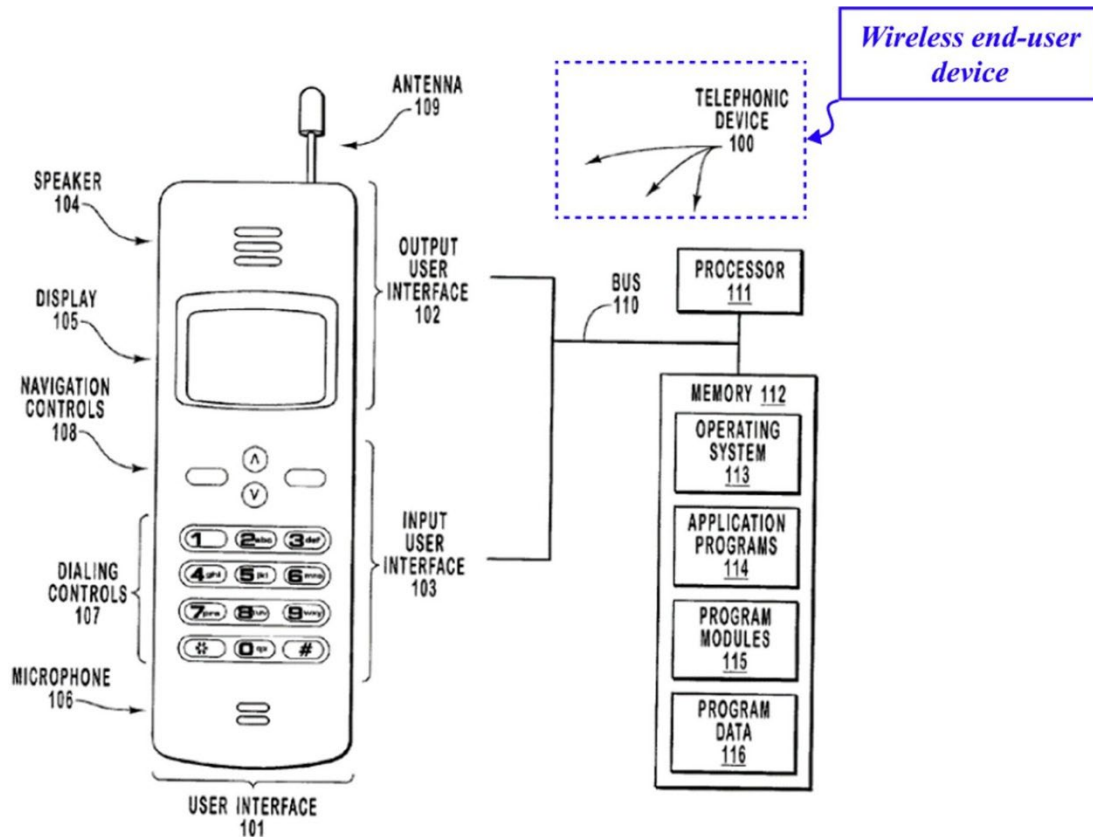


FIG. 1

EX1005, Fig. 1 (annotated).

84. The telephonic device 100 can be any device running a “suitable operating environment for the principles of the invention,” including “a general-purpose computing device.” EX1005, ¶38. Further, the telephonic device 100 may be implemented in “many types of computer system configurations,” such “personal

computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, and the like.” EX1005, ¶36. The telephonic device 100 also has “an antenna 109 if the telephone device 100 has wireless capabilities.” EX1005, ¶38.

2. [1a]: “**a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN;**”

85. In my opinion, Shell-Cole renders [1a] obvious based on disclosures in Shell and Cole regarding network connectivity. EX1005, ¶¶38, 55, 61; EX1006, 31, 35, 88, Fig. 2.

86. Shell discloses that telephonic device 100 includes various features that enable the device to establish a wireless network connection. EX1005, ¶¶20, 38. Shell instructs that a “wireless modem” is an example of how devices can establish a wireless connection. EX1005, ¶¶10, 13. Shell provides that “wireless carriers may define logic for accessing resources included in networks under their control[,]” suggesting connectivity of the telephonic device 100 with a cellular network (*wide area network*). EX1005, ¶55. This logic manages local resources, “such as a *modem* or network adapter” in the device, to establish connections using those resources. EX1005, ¶61. Although Shell explicitly discloses the use of modems, it does not

expressly disclose a WWAN modem. Nevertheless, Cole expressly discloses such a modem, “*WWAN modem 230*,” as shown below.

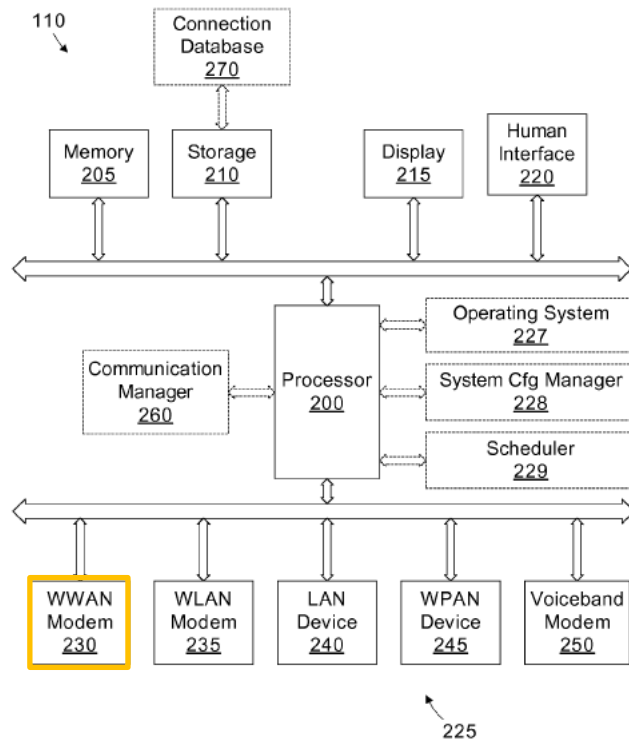


Figure 2

EX1006, Fig. 2 (annotated). Cole discloses a mobile device with different modems for wireless connectivity. In Cole’s Figure 2 (below), a mobile device includes a “WWAN modem 230” (*wireless wide area network (WWAN) modem*) providing connectivity to a “wireless wide area network (WWAN) 130,” confirming that such a modem configuration was well-known by the Critical Date. EX1006, ¶35.

87. Cole's "WWAN modem 230" is configured to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN. The "WWAN modem 230" can be a "3G channel" so that a mobile device can have network connectivity when in a "public place." EX1006, ¶88. Further, in Cole, a mobile device interfaces "with a plurality of communication networks 120, such as a wireless wide area network (WWAN) 130" (e.g., EVDO, HSDPA, 3G, GSM, CDMA, 2G, WiMAX, WiBro). EX1006, ¶31. As explained above in Sections X.B and XI.A, this modem configuration described in Cole represents implementation detail for Shell's telephonic device 100. EX1012, ¶¶95, 125. Thus, in Shell-Cole, Shell's telephonic device incorporates Cole's WWAN modem 230 and thereby would have enabled data to be wirelessly transmitted over a WWAN network.

88. Integration of "WWAN modem 230" from Cole into Shell's telephonic device 100 enables the Shell-Cole device to utilize data over a WWAN. As described in Cole, WWAN modem 230 is designed to interface with various communication networks such as EVDO, HSDPA, 3G, GSM, CDMA, 2G, WiMAX, and Wi-Bro. EX1006, ¶31. These networks provide the infrastructure necessary for high-speed data transmission and continuous connectivity. When Shell's device is configured with this modem, the device gains the ability to connect to these wide area networks, allowing the Shell-Cole device to send and receive data packets required for Internet

services like web browsing, email, and media content. This data communication is facilitated by the modem's capability to handle the necessary network protocols and manage the data flow between the device and the WWAN. EX1005, ¶¶6-8; EX1006, ¶88.

89. Moreover, Cole's network-selection techniques further enhance the device's ability to maintain robust Internet connectivity. EX1006, ¶6. Specifically, WWAN modem 230 enables the device to automatically select and connect to the most appropriate WWAN based on availability and signal strength. EX1006, ¶¶31, 39, 62. Such dynamic network selection ensures that the device remains connected to a WWAN, thereby providing continuous Internet service. The modem can function as a "3G channel" and is capable of maintaining connectivity even in public places. EX1006, ¶88. In Shell-Cole, this automatic selection and seamless switching between networks allows the device to consistently communicate data for Internet activities, ensuring that users can access online services reliably whenever the device is within the coverage of any supported WWAN. This capability, derived from Cole's detailed network-interface descriptions, illustrates how the combined disclosures of Shell and Cole render the Shell-Cole device's data communication functionality over a WWAN both feasible and obvious.

90. Further, accessing a network resource (as described in Shell) involves transmitting and receiving data over a network, examples of an *Internet service*

activity involving data exchange over the Internet. Examples of such Internet service activity in Shell include “electronic mail,” “a voice call,” “browsing stock quotes on the Internet,” and messaging via a “messaging application.” EX1005, ¶¶14, 15, 17, 57. Similar examples in Cole include “phone, email, or web” communications. EX1006, ¶¶80, 103-105. Indeed, the mechanisms and protocols used for accessing general network resources directly support Internet service activities, making them analogous in their operational principles and requirements. EX1017. A POSITA would have considered it obvious and conventional that the types of Internet service activities in Shell and Cole were analogous, interchangeable, and synonymous with each other for the types of mobile devices both Shell and Cole describe. Therefore, as discussed above, the Shell-Cole combination would have made it obvious for Shell’s telephonic device 100 to include **a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN.**

3. [1b]: **“a wireless local area network (WLAN) modem to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN; and”**

91. In my opinion, Shell-Cole renders [1b] obvious based on disclosures in Shell and Cole regarding network connectivity. EX1005, ¶35; EX1006, ¶¶31, 88, Fig. 2. In Cole’s Figure 2 (below), a mobile device includes a “WLAN modem 235”

(*wireless local area network (WLAN) modem*) providing connectivity to a “wireless local area network (WLAN) 140,” confirming that such a modem configuration was well-known by the Critical Date. EX1006, ¶35.

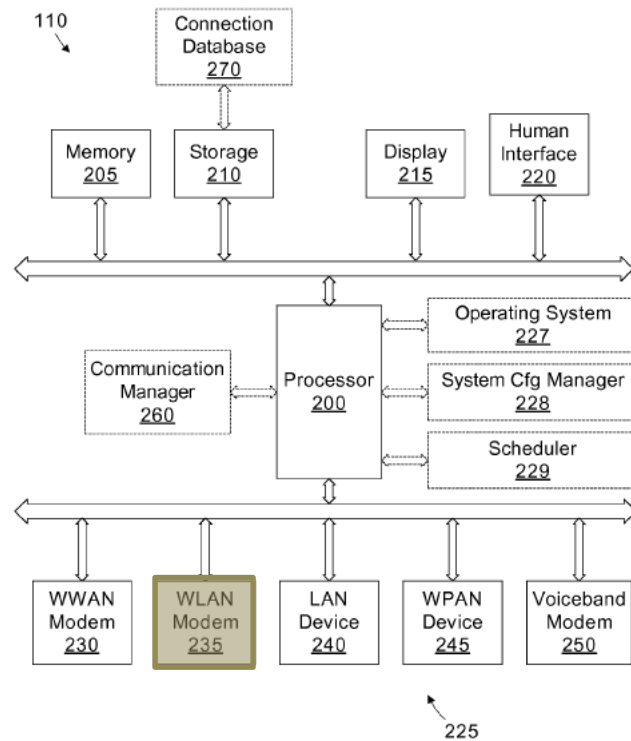


Figure 2

EX1006, Fig. 2 (annotated).

92. Cole discloses that its “WLAN modem 235” is configured to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN. The “WLAN modem 235” can be a “Wi-Fi connection” so that a mobile device can have network connectivity “[w]hen the user returns home.” EX1006, ¶88. Further, in Cole, a mobile device

interfaces “with a plurality of communication networks 120,” including a “wireless local area network (WLAN) 140” (e.g., IEEE 802.11, Wi-Fi). EX1006, ¶31. As explained in Sections X.B and XI.A, this modem configuration represents implementation detail for Shell’s telephonic device 100, which as discussed above discloses wireless connectivity but lacks specific descriptions of the modems that make these connections. EX1012, ¶¶95, 125. Thus, in Shell-Cole, Shell’s telephonic device incorporates Cole’s WLAN modem 235 and thereby would have enabled data to be wirelessly transmitted over a WLAN network. EX1012, ¶¶95, 125.

93. Integration of “WLAN modem 235” from Cole into Shell’s telephonic device 100 enables the Shell-Cole device to utilize data over a WLAN. In Cole, WLAN modem 235 is designed to interface with wireless local area networks, such as IEEE 802.11 and Wi-Fi. EX1006, ¶31. These networks provide the infrastructure necessary for high-speed data transmission and continuous connectivity while reducing network bandwidth costs. When Shell’s device is configured with this modem, the device gains the ability to connect to these local area networks, allowing the Shell-Cole device to send and receive data packets required for Internet services like web browsing, email, and streaming, without requiring access to a wide area network. This data communication is facilitated by the modem’s capability to handle the necessary network protocols and manage the data flow between the device and the WLAN. EX1005, ¶¶6-8; EX1006, ¶88.

94. Moreover, as discussed for [1pre], Cole's network-selection techniques further enhance the device's ability to maintain robust Internet connectivity by using different modems. EX1006, ¶6. This enables the device to be connected to a WLAN, thereby providing continuous Internet service. The modem can provide connectivity to a "home wireless network" or an "office wireless network," and is capable of maintaining connectivity with lower bandwidth costs. EX1006, ¶38. In Shell-Cole, this automatic selection and seamless switching between networks allows the device to consistently communicate data for Internet activities, ensuring that users can access online services reliably whenever the device is within the coverage of any supported WLAN. EX1006, ¶4. This capability, derived from Cole's detailed network interface descriptions, illustrates how the combined disclosures of Shell and Cole render the Shell-Cole device's data communication functionality over a WLAN both feasible and obvious. EX1006, ¶4. Further, as discussed for [1pre], accessing a network resource involves the transmission and reception of data over a network, which represents examples of an *Internet service activity* involving data exchange over the Internet. Examples of such Internet service activity in Shell include "electronic mail," "a voice call," "browsing stock quotes on the Internet," and messaging via a "messaging application." EX1005, ¶¶14, 15, 17, 57. Similar examples in Cole include "phone, email, or web" communications. EX1006, ¶¶80,

103-105. As discussed above, these conventional Internet service activities were well-known, interchangeable, and synonymous between the disclosures of Shell and Cole.

4. [1c]: “one or more processors configured to apply a stored WWAN differential traffic control policy to Internet data service provided using the WWAN modem and the at least one WWAN, and”

95. In my opinion, Shell-Cole discloses [1c] based on the processing capabilities of a mobile device taught in Shell. EX1005, ¶40, Fig. 1; EX1006, ¶34. In Shell, telephonic device 100 includes a *processor 111 (one or more processors)* with “complex and flexible general-purpose processing capabilities.” EX1005, ¶40. Cole likewise describes a mobile device 110 with a *processor 200*. EX1006, ¶¶15, 34, 91; *see* Section XI.B.1.

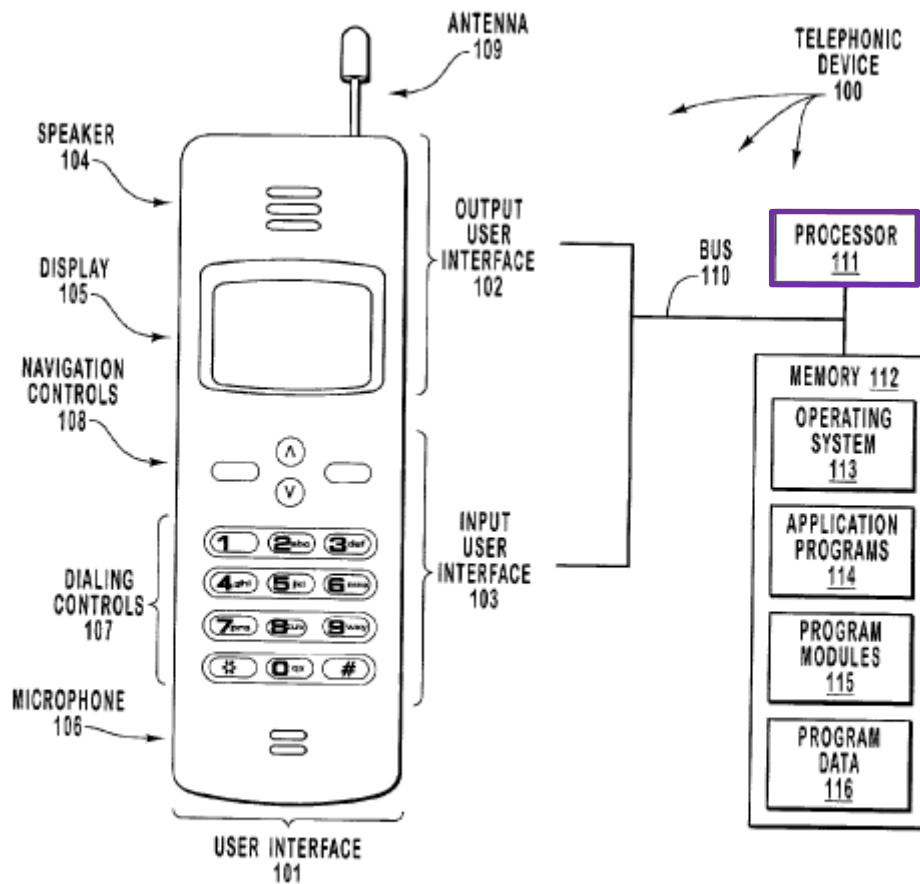


FIG. 1

EX1005, Fig. 1 (annotated).

96. Shell-Cole also renders obvious application of *a stored [] differential traffic control policy to Internet data service*, including as construed by the district court (*see* Section V), based on Shell's disclosures of "connection manager modules" regulating network access by applications on telephonic device 100. EX1005, ¶¶13, 17, 32; EX1006, ¶¶9, 80, 85. As discussed in Sections X.A and XI.A, in Shell, the "connection manager modules" are configured to regulate network access by applications, e.g., by prioritizing connection requests, determining an

appropriate method for establishing a network connection, granting a connection access request, etc. EX1005, ¶¶13, 17, 32. This is the function of a processor.

97. Specifically, a POSITA would have recognized that network connection (*Internet data service*) request handling is performed by Shell's "connection manager modules" and that functionality encompasses examples of applying a *traffic control policy* described in the '359 patent, such as "block/allow settings," "throttle settings," "adaptive throttle settings," and "QoS class settings." EX1001, 85:59-86:24. Further, as noted in the '359 patent, the "traffic control policy" can be used for "selecting the network from a list of available networks, blocking or reducing access until a connection is made to an alternative network, and/or modifying or replacing a network stack interface of the device to provide for intercept or discontinuance of network socket interface messages to applications or OS functions." EX1001, 85:59-86:24.

98. Like these examples, Shell teaches "connection manager modules" (or a "connection planner") receiving a connection request provided by an application that "request[s] access to a remote resource located in a destination network." EX1005, ¶33. The connection manager modules then "analyze[] a variety of factors to determine the most appropriate connection method to connect to the destination network." EX1005, ¶33. Further, "[w]hen the most appropriate connection method is known, a connection service provider associated with the connection manager may

cause a system level Application Programmer Interface ('API') to create a connection from a local computing device to the destination network." EX1005, ¶33. By granting or refusing connection requests, the connection planner is configured to apply decision logic (*stored [] differential traffic control policy*) for network connection (*Internet data service*) request handling (e.g., "block/allow settings"). EX1001, 85:59-86:24.

99. Further, in Shell, "connection requests may be prioritized so that higher priority requests cause lower priority requests to release a connection method so it may be used by the higher priority request." EX1005, ¶13. The "[p]riorities may be assigned to connection requests based on applications associated with the connection requests." EX1005, ¶57. This is shown in Figure 5A below:

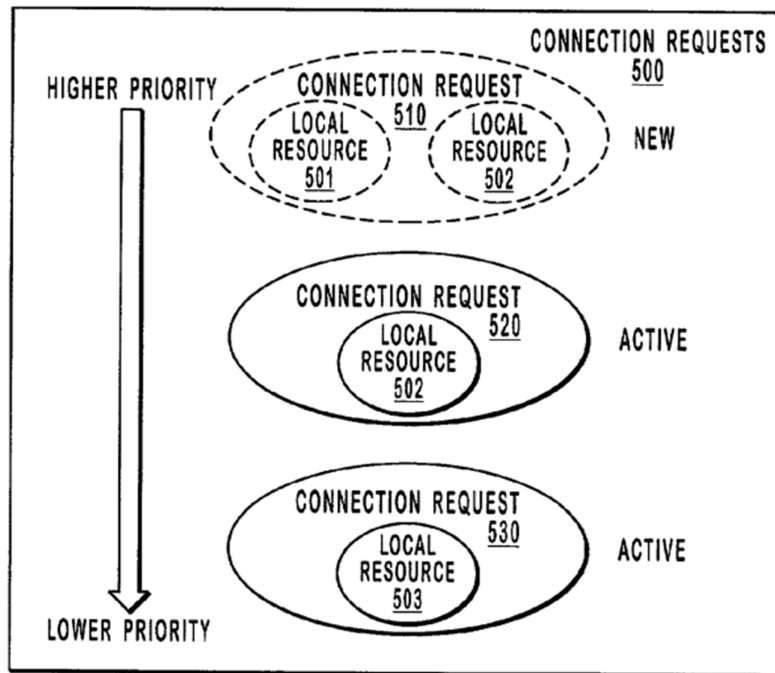


FIG. 5A

EX1005, Fig. 5A.

100. As shown above, “[c]onnection requests 510, 520, and 530 are prioritized, with connection 510 having the highest priority and connection 530 having the lowest priority.” EX1005, ¶57. A POSITA would have recognized that Shell’s prioritization techniques of network connection requests coincide with the ’359 patent specification’s disclosure of “blocking or reducing access until a connection is made to an alternative network” as an example of applying a “traffic control policy.” EX1001, 85:59-86:24. Indeed, Shell’s disclosures demonstrate “rule[s] for controlling network traffic that distinguish[] between two or more things” because the examples control access to a network.

101. Further, a POSITA would have found it obvious that decision logic applied by the Shell-Cole device would have provided *a WWAN differential traffic control policy* based on disclosures in Shell and Cole regarding regulating network access in relation to WWAN networks. Shell teaches multiple types of applications, e.g., data application(s), voice application(s), running on telephonic device 100 at the same time and submitting connection requests to an available network, such as a cellular network (e.g., WWAN) provided by a “wireless carrier.” EX1005, ¶¶49, 55. In this example, when a voice application submits a request to connect to the cellular network, the connection manager modules may prioritize the voice application over the data application. EX1005, ¶13. When this occurs, the data application module receives a notification that its “connection” is “being disconnected....” EX1005, ¶92. Thus, decision logic applied in this example relates to access to a cellular network and thereby involves applying a *WWAN differential traffic control policy*.

102. Similar to Shell’s decisional logic, Cole expresses that its “connection manager 260” may perform “communication load monitoring and deferred action functions,” which are “useful in many situations.” EX1006, ¶88. For example, if “the user is surfing a web site and selects a file for download[,]” “[t]he file [may be] larger than the user anticipated and is taking a long time to transfer.” EX1006, ¶88. In this example, “the lengthy transfer is conducted using the *WWAN modem 230* (e.g., a 3G channel), which consumes significant battery power[,]” and thus, “[t]he

connection manager 260 identifies the load on the WWAN modem 230 and asks the user to find a better connection.” EX1006, ¶88. As another example, where “a scheduled synchronization starts while the user is traveling[,]” the connection manager 260 may perform traffic monitoring to reduce “consum[ing] battery power.” EX1006, ¶89. The connection manager 260 “notices the load from the synchronization operation and asks the user if it should be deferred (i.e., no better connection is available).” EX1006, ¶89. Cole’s disclosures demonstrate that decisional logic, such the one described in Shell, are applicable to the allocation of resources (such as WWAN modems) and connections that supply Internet data to devices.

5. [1d]: “indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic control policy,”

103. In my opinion, Shell-Cole renders obvious [1d]. Indeed, as discussed below, Shell discloses this limitation in at least two ways. *First*, in response to an application’s connection request, Shell discloses its connection manager utilizes APIs to notify the application about the allocation of resources. *Second*, when an application makes a piggyback request, Shell discloses that the request and response to the request is handled through APIs associated with the connection manager.

a. Shell discloses its connection manager utilizes APIs to notify the application about the allocation of resources

104. In my opinion, Shell-Cole renders obvious [1d] because Shell discloses that its “connection manager modules 210 . . . may be configured as *APIs*” or are associated with “*APIs* . . . [that] may be called” (EX1005, ¶¶46, 48) and “may notify application modules 200” [i.e., *indicate to a particular application*] of a “notification associated with the connection to resources” [i.e., *one or more network access conditions*] after the connection manager “cause[s] resources to be released from and allocated to connection methods” [i.e., based on the applied *WWAN differential traffic control policy*] (EX1005, ¶92). See EX1005, ¶¶9, 22, 45-48, 70, 90-92.

105. As discussed above, Shell-Cole includes Cole’s *WWAN modem 230*, which would have made it obvious that the decision logic applied by the Shell-Cole device, in some circumstances, involves network connection handling relating to connection requests to access resources associated with a WWAN network (*WWAN differential traffic control policy*). EX1005, ¶¶13, 55, 92; EX1006, ¶¶15, 34, 91; Sections XI.B.1-XI.B.4.

106. Specifically, in Shell, “application modules may be notified” of a decision regarding a network connection request, and the “connection manager” may also “provide connection specific parameters” to the application modules. EX1005,

¶92. Thus, a POSITA would have recognized that, in Shell-Cole, processor 111 of telephonic device 100 is configured to provide **notification data** (e.g., decision, connection-specific parameters) related to a network connection request (*indicate*) to an application module that sent the network connection request (*particular application*).

107. The **notification data** (*one or more network access conditions*) includes the results of the decision logic. EX1005, ¶92. The notification data may relate to a decision regarding the network connection request, e.g., “a connection will be established,” “a connection is currently active,” “connection is being disconnected,” “no connection is currently active,” “a connection to a resource is blocked for some reason,” or “any other notification associated with the connection to resources” (each of which represents *one or more network access conditions*). EX1005, ¶92. As another example, the notification data may include “connection specific parameters,” e.g., “bandwidth, latency, power consumption, cost, physical location, past connection success, or other parameters” (each of which similarly represents *one or more network access conditions*). EX1005, ¶92.

108. Further, in Shell-Cole, the connection-manager modules “may be configured as **APIs**” or are associated with “**APIs** . . . [that] may be called. . . .” EX1005, ¶¶46, 48. When the connection manager is configured as an **API**, it “receive[s] standard data when being called by another module,” which simplifies

the establishment of connections. EX1005, ¶46. When the connection manager is associated with an **API**, this allows the connection manager to “be called instead of APIs associated with conventional protocol stacks.” EX1005, ¶48. In either instance, Shell’s teachings are designed so that the “connection manager operates between system level APIs and application programs, providing a level of abstraction.” EX1005, Abstract, ¶¶9, 22, 45-48, 70. Thus, functionality of the connection manager modules of Shell-Cole (e.g., providing notification data to application modules) may be implemented through use of well-known techniques for “providing a level of abstraction” between system-level APIs and applications. EX1005, ¶48. As one example, Irwin (EX1010), which was published on August 7, 2008, describes a Widget Engine 11 (with a connection manager 16) running on a mobile device and configured to call APIs relating to connection requests. EX1010, ¶¶13, 23, Fig. 2.

109. Specifically, Irwin discusses an example of a widget that calls a stock ticker function every hour that calls a function. “The function sends a request to a specific URL, parses the response and displays the latest share price to the user. When the timer fires, the resultant HTTP request will trigger the setup of a PDP context, thereby enabling the request and response to be sent at least partially over a wireless network.” *See* EX1010, ¶33. Figure 2 of Irving describes the code associated with a request for stock prices or a “getStockPrice() function.”

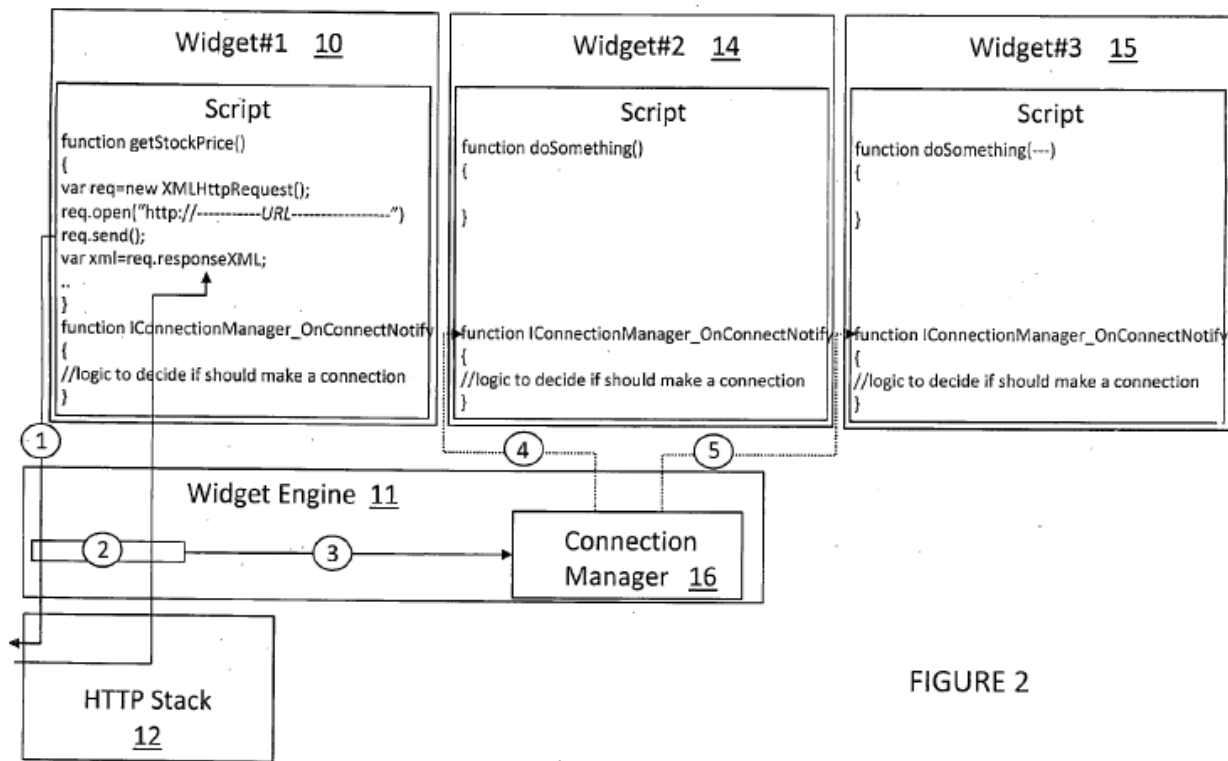


FIGURE 2

EX1010, Fig. 2. I have reproduced this code

```
var req = new  
  
XMLHttpRequest();  
  
req.open(http://-----URL-----);  
  
req.send();  
  
var xml=req.responseXML;
```

110. These lines of code represent the use of an API in order to enact the functionality discussed above. Namely, a data structure of type XMLHttpRequest() is created and assigned the variable name “req”. This object req can be used for further processing based on the API, with the “req.open()”

command creating a connection to a remote Internet destination and the “req.send()” apparently causing the request to be sent. A response from the destination is then collected into a variable “xml” through the API call req.responseXML. As these commands are not ones that would be exposed by an operating system, and do not represent functionality implemented directly within an application, they instead represent API calls associated with the creation and use of a network connection.

111. Further, mobile operating systems (e.g., Symbian OS) were known to include connection management services that used **APIs** to allow applications to access network connectivity information. EX1011, 45-46. Another example is EX1008 (Dive-Reclus), which ... uses permissions regulating network access by a third-party application using an API call. EX1008, 16:23-30, 17:1-28. Thus, in Shell-Cole, the connection manager modules would have been configured to provide notification data to application modules through use of well-known **APIs** used to provide network-related information (*via an application program interface (API)*). Indeed, Shell’s disclosures would have motivated a POSITA to use an API to communicate Shell’s notification data because this form of abstraction would have “reduce[d] the programming complexity associated with creation connections....” EX1005, ¶22. Moreover, a POSITA would have had a reasonable expectation of success in using an **API** to communicate with an application because this was a well-

known technique prior to the '359 patent and would have been within a POSITA's general knowledge.

112. Communicating network decisions to another application was a well-known means of providing information about authorization to access resources. There were many groups that attempted to provide application programming interfaces to access network functionality. Moerdijk (EX1016) described these efforts in 2003, writing that "different groups like Parlay [3] and JAIN [4,5] continued with efforts to develop application programming interfaces (APIs) that allow applications access to network functionality." EX1016, 1. Moerdijk further provides an overview of an API specification, including an API for "connectivity management." EX1016, 3.

113. Further, use of an API to communicate with an application was a well-known technique prior to the '359 patent and would have been within a POSITA's general knowledge. In my opinion, the use of an API does not make the '359 patent's claims unique from prior art as of the Critical Date, and a POSITA would have found it obvious to use an API to communicate Shell's network access conditions (e.g., connection request decisions and connection specific parameters) to applications. Additionally, this is particularly applicable when an application's connection request is blocked from accessing a network resource because the application is running in

the background (e.g., not interacting in the device display foreground with the user and needs some means of receiving an indication to adjust its operation).

b. Shell discloses that a piggyback request and response to the request is handled through APIs associated with the connection manager

114. In my opinion, Shell-Cole also renders obvious [1d] because Shell discloses that “sync engine 202 may wish to synchronize an electronic mail account [i.e., *a particular application*] only when a connection to the Internet has been established for another application. Applications that wish to piggy back, may call **APIs** associated [i.e., *via an application program interface (API)*] with connection manager modules 210 and be given an open handle,” which notifies the application (i.e., indicates to *a particular application*) of an established connection (i.e., *one or more network access conditions*). See EX1005, ¶¶70, 92. A POSITA would have understood or found it obvious that the notification communicating an established connection is based on the applied *WWAN differential traffic control policy* because its execution involves operation of the connection manager. See EX1005, ¶¶9, 22, 45-48, 70, 90-92.

115. As I discussed above (see Sections X.A-X.B & XI.A, XI.B.5), Shell-Cole’s connection manager applies a decision logic (*WWAN differential traffic control policy*) to determine which applications may utilize the combination’s WWAN modem and, using an **API**, notifies the requesting application about the

result of the decision logic (*indicate to a particular application // one or more network access conditions*). EX1005, ¶¶13, 55, 92; EX1006, ¶¶15, 34, 91. The use of Cole's *WWAN modem* in the Shell-Cole combination would have made it obvious that the decision logic applied by the Shell-Cole device, in some circumstances, involves network connection handling relating to connection requests to access resources associated with a WWAN network (*WWAN differential traffic control policy*). EX1005, ¶¶13, 55, 92; EX1006, ¶¶15, 34, 91.

116. Shell further suggests that these same teachings apply to Shell's piggyback requests as well. For example, Shell teaches: "Applications that wish to piggy back, may call *APIs* associated [i.e., *via an application program interface (API)*] with connection manager modules 210 and be given an open handle," which notifies the application of an established connection (i.e., *indicates to a particular application // one or more network access conditions*). See EX1005, ¶70. Open handles are not always given, however, as I detailed above with my discussion of EX1010 and below with EX1009. This notification is the exact type or at least mirrors the other notifications that Shell describes more generally. EX1005, ¶¶70, 93 ("application modules may be notified that a connection will be established...."). Moreover, because Shell's connection manager (via its connection planner 211) is responsible for processing the piggyback requests, the request is subject to Shell's decision logic (*WWAN differential traffic control policy*). EX1005, ¶70. This is

how Shell's piggyback requests are processed because the connection planner "evaluate[s] a variety of data when identifying a most appropriate connection method." EX1005, ¶56. In other words, the connection planner "execute[s] the [decision] logic" even to the piggyback requests (i.e., applies the *WWAN differential traffic control policy* to them). EX1005, ¶¶58-59, 68-70; *see also* EX1005, ¶¶9, 22, 45-48, 90-92. Indeed, this is consistent with how a POSITA would have understood Shell's piggyback request because network requests were known to not always be allowed.

117. Furthermore, APIs that allowed applications to learn of the unavailability of a connection were well known at the time of the Critical Date. The Java socket API discussed by Flanagan (EX1009) provides an example of this in code as a client application that connects to a server. A code comment describes functionality as "Create a socket to communicate with a server on port 6789 of the host that the applet's code is on. Create streams that use the socket. Then create a TextField for user input and a TextArea for server output. Finally, create a thread to wait for and display server output." EX1009, 3. A fragment of the code follows:

```
try {  
  
    s = new Socket(this.getCodeBase().getHost(), PORT); in = new  
    DataInputStream(s.getInputStream());  
  
    out = new PrintStream(s.getOutputStream());
```

```
...  
}  
  
catch (IOException e) this.showStatus(e.toString());
```

Within this code block, there is an API call to create a socket and some input and output functionality with the connected socket. If the socket fails to be established which could occur because of connectivity issues such as a blocked request, an error is reported back to the application in the form of an IOException. The details of the error are provided in text through the showStatus function.

6. [1e]: “wherein the **one or more network access conditions** includes **a network access condition** that indicates the **unavailability** to the **particular application** of an Internet data service that is currently **available** via the **WWAN modem** to a **different application**, but that is indicated as **unavailable** to the **particular application**.”

118. In my opinion, as discussed in [1d], Shell-Cole renders obvious [1e]. Indeed, Shell discloses this limitation in at least two ways. *First*, in response to an application’s connection request, Shell discloses its connection manager utilizes APIs to notify the application about the allocation of resources, indicating the availability or unavailability of a resource to the application. *Second*, a POSITA would have found it obvious that the responsive notification to Shell’s piggyback request involves an indication of the availability or unavailability of a resource to the application.

a. Shell discloses allocating a resource to one application but not another based on the priority of the application's connection request and notifying the application about the availability of the resource

119. In my opinion, Shell-Cole renders obvious [1e] because Shell teaches that “telephonic device 100 may request synchronization of an electronic mail account [i.e., a *different application*] and access to a remote web page [i.e., a *particular application*],” using the same resource (i.e., *WWAN modem*) and connection (i.e., *Internet data service*), but teaches that the connection manager may be configured to allocate resources to the synchronization of an electronic mail account based on priority. EX1005, ¶¶15, 17, 57, 67-68. As with the discussion in [1d] above (*see* Section XI.B.5.a), when the connection manager allocates resources, it also notifies the requesting application about the availability of the resource and connection. *See* Section XI.B.5.a. Accordingly, a POSITA would have found [1e] obvious based on Shell's disclosures.

120. Shell contemplates allowing some applications to use a connection while preventing other applications from using the same connection. As discussed above, Shell-Cole's connection manager executes decisional logic (i.e., *differential traffic control policy*) to allocate resources to applications based on priorities. *See* Sections XI.B.4-XI.B.5. When resources are allocated or not allocated, the connection manager notifies the application of the decision (i.e., *indicates to a*

particular application). See EX1005, ¶92 (“For example, application modules may be notified that a connection will be established, that a connection is currently active, that connection is being disconnected, that no connection is currently active, that a connection to a resource is blocked for some reason, or any other notification associated with the connection to resources.”). Shell further discloses that “there may be a plurality of new connection requests at a given time.” EX1005, ¶¶15, 68. In other words, “telephonic device 100 may request synchronization of an electronic mail account and access to a remote web page,” using the same resource and connection. EX1005, ¶¶15, 68. Or “a voice call [i.e., a *different application*] may be given higher [] priority over data calls [i.e., a *particular application*] that share the same modem.” EX1005, ¶13. However, Shell also provides that just because both requests are made does not mean that both will obtain access to the resource. EX1005, ¶¶13, 17, 67-68. For example, the “connection manager may delay the request to browse the Internet until mail synchronization has completed, or take some other action to ensure that the lower priority request does not adversely impact the high priority request.” EX1005, ¶¶17, 67. Shell also discloses that other actions beyond delay may include blocking the application from accessing the connection. EX1005, ¶92.

121. Shell’s decision logic further describes how multiple application connection requests would be handled in Figures 5A and 5B. EX1005, ¶¶57-67,

Figs. 4, 5A, 5B. Figure 5A depicts a prioritization order of connection requests 510, 520, and 530 while Figure 5B illustrates the resources (e.g., modems) available to make the connections. EX1005, ¶57.

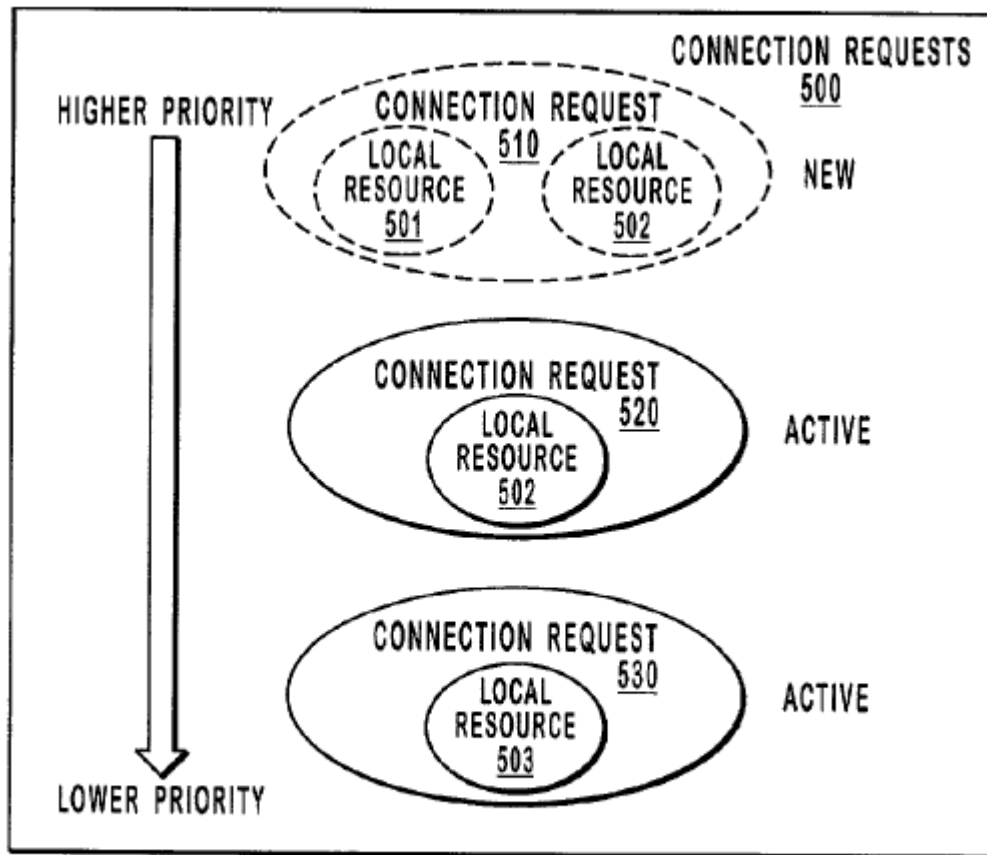


FIG. 5A

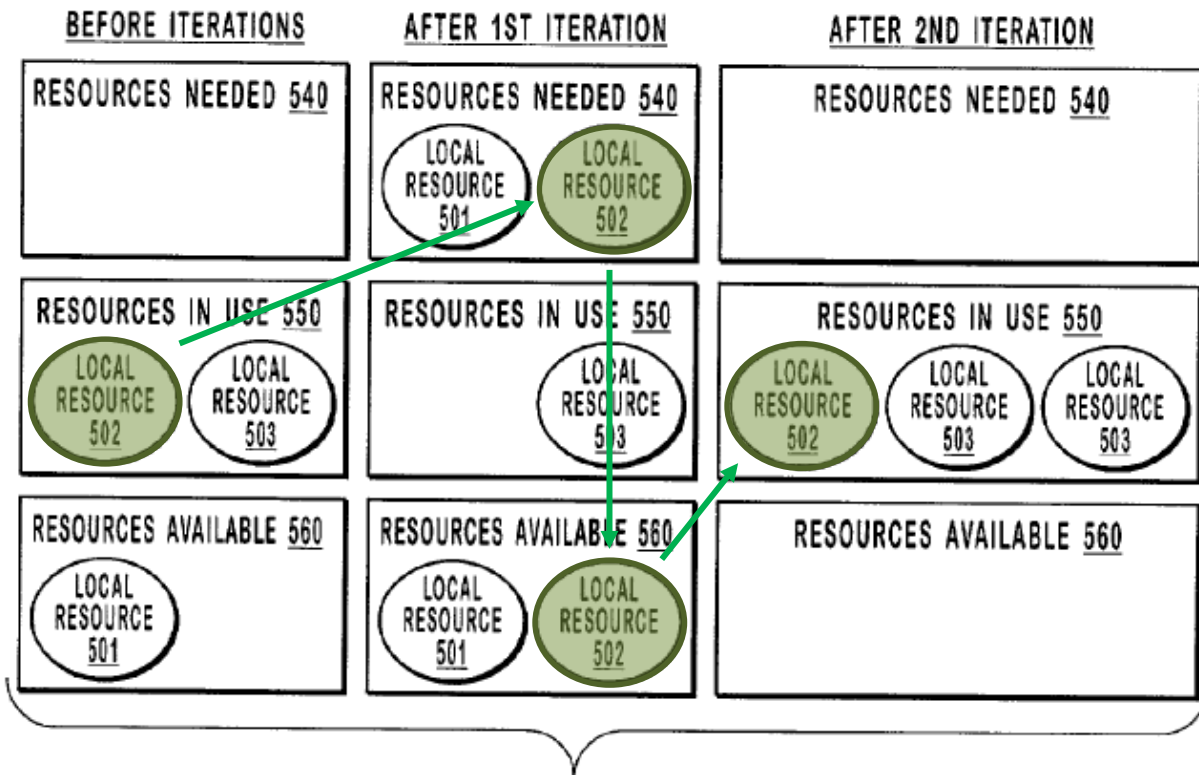


FIG. 5B

EX1005, Figs. 5A, 5B (annotated).

122. At the start of the process (“Before Iterations”), resources 502 and 503 are in use (by connection requests 520 and 530), and resource 501 is available. EX1005, ¶60. In the first iteration, when a new connection request 510 is made (e.g., when a user requests the synchronization of an electronic mail account [i.e., a *different application*]), the connection manager registers that the new connection request 510 needs local resources 501 and 502. EX1005, ¶¶61-66. As the process continues, the connection manager weighs the priority of the new request 510 against the active connections 520 (e.g., a remote web page [i.e., a *particular application*]),

and 530 to determine how to allocate the resources. EX1005, ¶67. Because new request 510 has a higher priority than request 520, the connection manager disconnects the application associated with request 520 to resource 502. EX1005, ¶67. The connection manager then allocates resources 501 and 502 to new request 510, which provides the application associated with new request 510 with a connection. EX1005, 67. The above process describes that resource 502 is made *unavailable* to connection request 520 but is *currently available* to new request 510. EX1005, ¶67. A POSITA would have understood that this allocation of resources is exactly what is claimed in [1e]. And indeed, such an allocation of resources is consistent with the '359 patent's own description. *See* EX1001, 101:47-102:56 (describing that “the application traffic prioritization engine 2710 uses a policy stored in the network service usage control policy datastore 2712 to determine an appropriate prioritization for traffic to and/or from the network service consuming application 2702.”). The '359 patent's “control policy,” just like Shell-Cole, can “throttle network resource consumption, prohibit the use of network resource[s] by certain applications, etc.” EX1001, 101:55-57.

123. As discussed above, Shell further discloses that when the connection manager releases or allocates resources, it “may notify application modules 200 of such changes.” EX1005, ¶92. Shell explains that “application modules may be notified that a connection will be established, that a connection is currently active,

that connection is being disconnected, that no connection is currently active, that a connection to a resource is blocked for some reason, or any other notification associated with the connection to resources.” EX1005, ¶92. In other words, when a resource is made unavailable to one application and available to another application, these applications are each notified of the availability of the connection. In my opinion, this encompasses a notification to one application that a connection is unavailable to that application while that same connection is currently available to a second application. For example, the application that sent connection request 520 would receive a notification that resource 502 and the connection is unavailable even though resource 502 is currently available to the application corresponding to request 510. I further believe that a POSITA would have found it obvious to send availability notifications based on Shell’s notification disclosures. For example, a POSITA would have understood that a notification that a connection will be established/is currently active is an indication of availability because these connections allow an application with access to the Internet. A POSITA would have further understood that a notification that a connection to a resource is blocked is an indication of unavailability because the application is prevented from accessing the Internet.

124. Based on these disclosures, a POSITA would have understood that when the processor of Shell-Cole’s device reallocates resources, it notifies (i.e.,

indicates) the *unavailability* of a connection using resource 502 (i.e., *Internet data service via the WWAN modem*) to the application associated with request 520 (i.e., *particular application*) even though the connection using resource 502 is currently *available* to the application associated with new request 510 (i.e., *different application*). Moreover, a POSITA would have found it obvious to use a WWAN over other types of networks. A POSITA would have been motivated to use a WWAN over other applications at least when other networks are unavailable due to a user's location (e.g., out of range of a WLAN). EX1007, ¶4. Indeed, using a WWAN would have provided a user with flexibility that other networks may not provide. EX1007, ¶4. A POSITA would have had a reasonable expectation of success in using a WWAN for applications because it was known that WWANs support the transmission of Internet data. EX1007, ¶4. Indeed, as discussed in [1c] (Section XI.B.4), Cole even suggests the use of decisional logic to allocate resources (e.g., a WWAN modem) that results in a connection being available or unavailable to an application. EX1006, ¶¶88-89.

b. A POSITA would have found it obvious that the responsive notification to Shell's piggyback request involves an indication of the availability or unavailability of a resource to the application

125. In my opinion, Shell and Cole further render obvious [1e] because Shell teaches that "an application [i.e., *particular application*] may request to only access

resources in a destination network when another application [i.e., *different application*] is already accessing resources.” EX1005, ¶70. Shell’s piggyback requests “may call **APIs** associated with connection manager modules 210 and be given an open handle” that notifies the application whether a connection has been established. EX1005, ¶70. As noted above, Shell instructs that its communication manager notifies applications of, for example, a “notification associated with the connection to resources [i.e., the *availability* or *unavailability* to the applications].” EX1005, ¶92. In other words, Shell describes how resources that establish connections (i.e., *a WWAN modem that provides an Internet data service*) may be available to some applications but not others. EX1005, ¶¶70, 92. Because Shell discloses allocating a resource to one application while blocking the same resource to another application, a POSITA would have found it both disclosed and obvious to block piggyback requests as well.

126. Shell describes that a piggyback request occurs when one application requests access to resources that another application is already using. EX1005, ¶70. “For example, sync engine 202 may wish to synchronize an electronic mail [i.e., a *particular application*] account only when a connection to the Internet *has been established* for another application [i.e., a *different application*].” EX1005, ¶70. Indeed, email synchronization may be a particular or different application based on whether “a user specif[ies] the synchronization by activating a control in a user

interface.” EX1005, ¶14. These disclosures reveal that Shell contemplates that certain applications (and their requests for network connection), such as email, may be initiated by the user or done automatically. And, therefore, an application could be a particular or a different application depending on the context how the application is being used. Shell emphasizes that a piggyback request is an attempt to use the same connection that is already being used, which is contrasted from instances where “[a] variety of connection methods may be used.” EX1005, ¶70; *see also* EX1005, ¶68. In other words, a piggyback request involves a request for *an Internet data service that is currently available via the WWAN modem to a different application.*

127. Shell describes notifying the piggyback requesting application of an established connection (i.e., *indicates to a particular application // one or more network access conditions*). *See* EX1005, ¶70. Further, as discussed for [1d] (*see* Section XI.B.5.a), as with any other connection request, Shell’s connection planner “execute[s] the [decision] logic” to the piggyback requests (i.e., applies the *WWAN differential traffic control policy* to them), which then triggers the allocation of resources to the applications and notifications of the allocations. EX1005, ¶¶58-59, 68-70; *see* Section XI.B.5.a. In my opinion, a POSITA would have understood that, from the piggyback-requesting application’s standpoint, the request is similar to any other connection request. This is because at the end of the

day a piggyback request is a connection request—only that a piggyback request does not occur until another application has established a connection. Consequently, a POSITA would have known that the notifications associated with the connections discussed above also would have been applied to the piggybacking requests. These notifications would include instances when a resource is not allocated and “a connection to a resource is blocked for some reason.” EX1005, ¶92. As discussed above, when a resource is not allocated and/or a connection is blocked, this signifies that the Internet is unavailable to the application. Indeed, a POSITA would have found this to be the case because blocking a connection from being established prevents the application from accessing the Internet. Shell suggests, and a POSITA would have found it obvious, that these same allocation/notification teachings advantageously apply to piggyback requests as well. For example, because Shell’s decision logic is invoked for piggyback requests (*see* Section XI.B.5.a), a POSITA would have understood that the notification response to piggyback requests may include indications to the requesting application of the *availability/unavailability* of a resource and connection for the same reasons discussed in Section XI.B.5.a.

128. A POSITA would have found it obvious and beneficial that Shell-Cole’s connection manager would have applied the same decision logic to piggyback requests that Shell discloses for independent requests based on Shell’s disclosures. A POSITA would have been motivated to apply the same decision logic to

piggyback requests because the purpose of the connection manager is “to simplify the amount of data a user must be familiar with in order to request a connection to a remote resource.” EX1005, ¶¶32, 11-12. Indeed, applying the same decision logic and notification responses to piggyback requests would have furthered this goal as it would prevent a user from having to engage with the connection request by selecting which network to use. A POSITA would have had a reasonable expectation of success in implementing the decision logic to piggyback requests because they are already processed by the connection manager. Running a piggyback request through the decision logic would merely have involved the same routine programming, as expressly disclosed and encouraged, that a POSITA would have used to process other connection requests from applications.

129. In particular, a POSITA would have found it obvious to deny a piggyback request and indicate to the requesting application that the resources and connection are unavailable to the application. For example, a POSITA would have been motivated to deny a piggyback request because providing the requesting application with access to the resource and connection may risk exceeding the allocated bandwidth of the connection and negatively impacting the performance of the application already using the resource/connection. Indeed, a POSITA would have found denying piggyback requests to be consistent with Shell’s teachings that lower priority requests may be delayed to ensure that the request “does not adversely

impact the high priority request.” EX1005, ¶17. A POSITA would have had a reasonable expectation of success in the modification for the same reasons mentioned above, and because Shell expressly sets forth its decision logic to be applied to piggyback requests.

C. **Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”**

130. In my opinion, Shell-Cole renders claim 26 obvious because Shell teaches prioritizing the allocation of resources and connections for active applications (i.e., a *different application*) over other applications based on a user “[a]ctivating a control in a user interface” (i.e., *determine that the different application is active based on user interaction with the different application*). EX1005, ¶¶13-14, 51, 57, 92.

131. In Shell-Cole, a user’s interaction with an application may instigate connection requests, allowing its **processor** to determine that the application is active. EX1005, ¶14, 51. For example, to connect to a network, Shell describes that “a user may request access to a resource that is accessed by connecting to a destination network.” EX1005, ¶14. Shell discloses that “[t]his may be as simple as a user specifying the synchronization of an electronic mail account by activating a control in a user interface.” EX1005, ¶¶14, 51. Indeed, it is the user’s interaction

with the application that instigates the application to send a connection request, which triggers the decision logic that subsequently establishes a network connection based on priority. *See* Sections X.A-X.B & XI.A. A user may also place a voice call that then causes the connection manager to execute its decision logic and determine whether the voice call should be allocated resources over a data call that would share the same modem. EX1005, ¶¶8, 13. A POSITA would have understood a user's input may instigate a regular connection request (Section XI.B.6.a) and may also serve as the foundational connection request upon which a secondary application attempts to piggyback (Section XI.B.6.b).

132. As discussed above in [1e], Shell describes that connection-manager modules may assign priorities to network-connection requests in a variety of ways. EX1005, ¶57; Sections XI.B.6.a-XI.B.6.b). This includes assigning priorities at least “based on the current state of applications associated with the connection requests;” for example, “a request from an application that is currently active may be assigned a higher priority than a request from an application running in the background.” EX1005, ¶¶57, 92. And the currently active requests include those initiated by a user. EX1005, ¶57.

133. Based on these disclosures, a POSITA would have understood or would have found it obvious that Shell's **processor** is configured to assess (i.e., determine) whether a request is coming from an application that the user is interacting with as

part of the prioritization of connection requests as I discussed in Sections XI.B.6.a and XI.B.6.b. EX1005, ¶¶51, 57, 92. This is because user input is a way of activating an application, and in certain circumstances like a phone call or web browsing, the processor would only provide these applications with an Internet connection when they are in use. *See* EX1005, ¶51 (“Activating a control in a user interface, for example, a button, a drop down list or other element, may initiate such requests.”); EX1005, ¶¶57, 92. Accordingly, a POSITA would have known and considered obvious that the assessment of whether a request is coming from an in-use application is a determination that the different application is an active application based on a user interaction with the different application.

134. As discussed in Sections XI.B.5.a and XI.B.6.a, Shell describes that when a new connection request 510 is made (e.g., when a user requests the synchronization of an electronic mail account or a user places a voice call [i.e., a *different application*]), the connection manager registers that the user interacted with the email application, which initiated the request. EX1005, ¶¶13-14, 61-66. The new connection request 510 asks for local resources 501 and 502. EX1005, ¶¶61-66. And as the process continues, the connection manager weighs the priority of the new request 510 against the active connections 520 (e.g., a remote web page or a data request [i.e., a *particular application*]), and 530 to determine how to allocate the resources. EX1005, ¶¶13-14, 67. Because new request 510 has a higher priority than

request 520, the connection manager disconnects the application associated with request 520 to resource 502—for example, because the email application that is currently active while the web page is operating in the background. EX1005, ¶¶13-14, 67. The connection manager then allocates resources 501 and 502 to new request 510, which provides the application associated with new request 510 a connection. EX1005, 67. The above process describes that resource 502 is made *unavailable* to connection request 520 but is *currently available* to new request 510. EX1005, ¶67. Shell-Cole therefore describes or renders obvious that a user's input or interaction with an application may be part of the decision logic and taken into account in the determination of a priority assignment.

135. As a result, applications that are currently active based on user interaction with the application (e.g., synchronizing email or placing a voice call) are connected to a network while other applications that are running in the background are not provided with access to the network. As discussed in Sections XI.B.6.a and XI.B.6.b, Shell-Cole's device monitors the allocation of resources and connections and notifies the applications about the status of their requests. As applied to the decision flow described in Figures 5A/5B of Shell, an application that is operating in the background (i.e., a particular application), would be notified of the reallocation of resources and connection status while the same resources and connection are currently available to the active application. Therefore, a POSITA

would have found claim 26 obvious based on the expressly disclosed decision flow of Shell used in the device of Shell-Cole.

136. In my opinion, and as discussed in Sections XI.B.5.b and XI.B.6.b, the same decision logic and notifications of Shell-Cole apply to piggyback requests. A POSITA would have recognized that a user's input can initiate a regular connection request (Section XI.B.6.a) and may also serve as the foundational connection request upon which a secondary application attempts to piggyback (Section XI.B.6.b). *See* EX1005, ¶70 (“an application [i.e., *particular application*] may request to only access resources in a destination network when another application [i.e., *different application*] is already accessing resources.”). Consequently, a POSITA would also have found claim 26 obvious based on the disclosed techniques of piggybacking in the device of Shell-Cole.

XII. GROUND 2: SHELL, COLE, AND FLACK RENDER OBVIOUS CLAIM 1

137. In my opinion, Shell and Cole further in view of Flack also renders obvious claims 1.

A. [1pre]-[1d]:

138. In my opinion, Shell-Cole renders obvious [1pre]-[1d] for the same reasons discussed above in Ground 1. *See* Sections XI.B.1-XI.B.5.

- B. [1e]: “wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”**

139. In my opinion, to the extent there is a question that Shell-Cole discloses or renders obvious [1e] based on Shell-Cole’s denial of an application’s piggyback request (Sections XI.B.5.b and XI.B.6.b), a POSITA would have further found [1e] obvious because Flack teaches that a “first program may be blocked from accessing the network based on network connectivity that has already been established for the second program.” EX1007, ¶5.

140. As discussed above (Section X.C), Flack discloses a system that allows applications access to a network based on a communications configuration. EX1007, 20. Flack instructs that its system includes a processor that executes its process. EX1007, ¶¶39, 63. As part of this process, the client “identifies an application associated with the communications activity (420). Identifying an application may include identifying the source of a communications activity ... Some of these applications may represent user applications that a user has launched in attempting to retrieve information from the Internet (e.g., a web browser, an email application, or an Instant Messaging application).” EX1007, ¶50. Flack discloses that “[i]t might be better to establish and preserve connectivity for only selective user applications,” rather than applications/programs that operate in the background. EX1007, ¶16.

Flack teaches that there are two ways to respond to a connection request from a first program to use the same “network connectivity that has already been established for [a] second program.” EX1007, ¶¶5-6, 42, 57-58, Figs. 4, 4A. These include “enabl[ing] access to the network” and “block[ing] access[]” to the network. EX1007, ¶¶5, 58.

1. Motivation to Combine Shell, Cole, and Flack

141. Shell, Cole, and Flack each relate to wireless devices with similar functions and uses. For example, Shell, Cole, and Flack all detail wireless devices that are configured to connect to different networks. EX1005, ¶13, EX1006, ¶35, EX1007, ¶44. Shell, Cole, and Flack describe methods that involve the use of policies to determine whether an application should be connected to a network. EX1005, ¶57, EX1006, ¶36, EX1007, ¶5.

142. Given Shell, Cole, and Flack’s similarities, a POSITA would have been motivated with a reasonable expectation of success to consider their combined teachings in order to optimize network usage, battery and power consumption, and cost. Aiming to “establish[] and maintain[] connections in a manner that promotes cost efficiency and efficient utilization of available connection methods,” Shell describes a connection manager that enables a user to merely “request the resource he or she wishes to connect to and the connection manager devices which connection method to utilize.” EX1005, ¶93; *see also* EX1005, ¶¶1, 12-13. Shell suggests that

this may be simple when “a user specif[ies] the synchronization of an electronic mail account by activating a control in a user interface.” EX1005, ¶14. But this may be more complicated “when a user attempts a voice call on a telephone line that is currently being used for a data connection.” EX1005, ¶14. Shell provides other examples of when multiple applications request access to the same resource, such as piggybacking, when “sync engine 202 may wish to synchronize an electronic mail account only when a connection to the Internet has been established for another application.” EX1005, ¶70. Although Shell suggests that these requests can be made and blocked, Shell does not expressly discuss every technique for efficiently handling piggyback requests. EX1005, ¶70. A POSITA would thus have looked to other references to better understand the outcomes of a piggyback request and to efficiently handle them under different network and operational circumstances.

143. A POSITA would have looked to and appreciated Flack for these details because Flack discusses balancing network access for multiple applications. As noted above, Flack describes that “[i]t might be better to establish and preserve connectivity for only selective user applications,” rather than applications/programs that operate in the background. EX1007, ¶16. Flack suggests two ways in which a processor may respond to a connection request when a second program is already running: “enable[ing] access to the network” and “block[ing] access[.]” to the network. EX1007, ¶¶5-6, 42, 57-58, Figs. 4, 4A.

144. A POSITA would have been motivated to implement Flack's specific allowing/blocking decisions into the processing of Shell-Cole's piggyback request because doing so would further Shell's goal of promoting "cost efficiency and efficient utilization of available connection methods." EX1005, ¶93. For example, blocking access to a background application's update/synchronization as Flack suggests would "ensure that the lower priority request does not adversely impact the high priority request" as Shell provides. EX1005, ¶17. This would have been an efficient utilization of available connection methods and resources because it would prevent the overloading of a connection method. EX1006, ¶¶9, 74. Furthermore, it would have reduced the risk of latency in active applications. EX1006, ¶¶9, 74. Moreover, as described in Ground 1 [1d] and [1e], Shell-Cole provides notification data (i.e., indicates a network access condition) to applications as part of its process. A POSITA would have included Flack's specific **allowing/blocking** decisions in Shell-Cole's notification data because this information informs the requesting application whether it has been allocated resources, allowing the application to know whether it can execute a task that prompted the request.

145. A POSITA would have had a reasonable expectation of success in implementing Flack's blocking step into Shell-Cole because Shell already is configured to block certain connection requests. EX1005, ¶92. Shell-Cole already includes the physical components to process application data for network requests

(see Sections X.A-X.B, XI.A, XI.B.2-XI.B.4); therefore, a POSITA would have understood that implementing Flack's teachings into Shell-Cole would merely have required routine programming that was well within a skilled artisan's toolbox prior to the earliest claimed priority date. Moreover, as in the Shell-Cole combination, Flack discloses that this system may be implemented on a cellphone communicating on a WWAN, such as a "wireless circuit-switched network." EX1007, ¶15. A POSITA would have thus understood that Flack is applicable to WWANs/WWAN modems because it suggests compatibility with cellular networks. Combining the references would have been nothing more than applying a known technique—i.e., incorporating Flack's denial of connection requests—to a similar mobile device communications system—i.e., Shell-Cole's system. The implementation would have yielded predictable results: denying Internet access to an application that is not actively being used and notifying the application of the denial. And the result would have been advantageous and obvious enhancements in network bandwidth usage, power consumption, and application activity.

XIII. GROUND 3: SHELL, COLE, AND FLACK RENDER OBVIOUS CLAIM 26

- A. **Claim 26: “The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”**

146. In my opinion, Shell-Cole (Ground 1) further in view of Flack and Shell, Cole, and Flack (Ground 2) also render obvious claim 26 because Flack discloses that when its “client determines that communications activity [within an application] **represents user actions**,” then the client may establish a network connection. EX1007, ¶19, *see also* ¶¶8, 16-17, 50.

147. As discussed above, Shell teaches that “a request from an application [i.e., a **different application**] that is currently active may be assigned a higher priority than a request from an application [i.e., **particular application**] running in the background.” EX1005, ¶¶57, 92. Shell notes that requests from the application modules may come from a user “[a]ctivating a control in a user interface.” EX1005, ¶¶14, 51. In other words, Shell describes that the request may be made based on a user’s interaction with the device allowing its **processor** to determine that the **different application** is active. EX1005, ¶¶14, 51.

148. In the same types of mobile devices as Shell, Flack teaches allowing applications access to a network based on a communications configuration. EX1007, ¶20. Flack instructs that the client includes a processor that executes its process.

EX1007, ¶¶39, 63. As part of this process, the client “identifies an application associated with the communications activity (420). Identifying an application may include identifying the source of a communications activity Some of these applications may represent user applications that a **user has launched in attempting to retrieve information from the Internet** (e.g., a web browser, an email application, or an Instant Messaging application).” EX1005, ¶50. “If the client determines that communications activity represents **user actions**, the client may establish network connectivity by, for example, establishing a dialup connection, so that the client may support the user activity within the user application.” EX1007, ¶19; *see also* EX1007, ¶¶8, 16-17, 19, 50. Further, as noted above in Sections X.C and XII.B, Flack describes how certain applications may be allowed or blocked access to a network if another application is already using the network connection. EX1007, ¶¶5, 50. Further to what I discussed in Ground 1, Flack further evidences that a POSITA would have understood a user interfacing with an application may instigate a regular connection request (Section XI.B.6.a) and may also instigate the foundational connection request upon which a secondary application attempts to piggyback (Section XI.B.6.b). *See* EX1007, ¶5 (“The first program may be blocked from accessing the network based on network connectivity that has already been established for the second program.”).

149. A POSITA would have found it obvious to implement Flack's determination of application status based on **user activity** as part of Shell-Cole's traffic policy related to regular connection requests (Ground 1, Sections XI.B.5.a, XI.B.6.a, XI.C) and piggyback requests (Ground 2, Section XI.B.5.b and XI.B.6.b). For example, a POSITA would have implemented Flack's "determin[ation] that communications activity represents user actions" as part of Shell-Cole's decision logic that makes priority determinations based on, in-part, "a request from an application that is currently active ... [and] a request from an application running in the background." EX1005, ¶57. This combination would provide resources and network connections to applications that are currently active based on user interaction with the application (e.g., synchronizing email or placing a voice call) while preventing resource and network access to other applications that are running in the background. As discussed in Sections XI.B.6.a and XI.B.6.b, Shell-Cole's device monitors the allocation of resources and connections, and notifies the applications about the status of their requests; the addition of Flack would not modify Shell-Cole's notification steps. Indeed, as applied to the decision flow described in Figures 5A/5B of Shell, an application that is operating in the background (i.e., a **particular application**), would be notified of a change in availability of resources and connection status as a consequence of the reallocation of resources and connection status, while the same resources and connection are

currently available to the active application. Therefore, a POSITA would have found claim 26 obvious based on the expressly disclosed decision logic of Shell-Cole, which processes user-initiated connection requests, together with Flack's determination of application status based on **user activity**.

150. A POSITA would have further found it obvious to implement Flack's determination of application status based on user activity as part of Shell-Cole-Flack's traffic policy as discussed in Ground 2 (Section XII.B). For example, as described above (Sections XI.B.5.b, XI.B.6.b, XII.B), the same decision logic and notifications that apply to other connection requests also apply to piggyback requests. Consequently, a POSITA would have also found claim 26 obvious based on the disclosed technique of piggybacking as well.

151. A POSITA would have been motivated to implement both changes for the same reasons. In particular, a POSITA would have been motivated to do so because users would have wanted the application that they are currently using to be connected to the Internet (when the application requires a data connection to operate). Indeed, without an Internet connection, some applications, such as a web-based application, would have been unusable; consequently, a POSITA would have wanted to prioritize providing Internet data to active applications over inactive or background applications. A POSITA would have had a reasonable expectation of success in this combination because it would have involved minor programming

changes to Shell's prioritization preferences. The addition of Flack's user interaction decision would apply equally to the processing of a regular connection request and a piggyback request. Indeed, Shell already contemplates that "[p]riorities may also be assigned to connection requests based on the current state of applications associated with the connection requests." EX1005, ¶57. And, therefore, prioritizing based on the active state of an application would have merely been one of these possible states. A POSITA would have also found that this modification would have been consistent and compatible with Shell's teachings that a "variety of connection methods may be used" to satisfy different connections request. EX1005, ¶¶69, 17. This is because a WWAN-based connection might be the best option for certain applications at a specific time when the user is actively using the application while a WLAN-based connection might be more appropriate for other applications at the same time. EX1006, ¶¶4, 64. As an example, using a WWAN-based connection to download a document may not be an appropriate connection because it could incur extra charges; however, a WLAN may not incur the same charges. EX1005, ¶6; EX1006, ¶88.

152. In my opinion, Shell-Cole-Flack renders obvious claim 26. Flack's mechanism for determining user-driven communications activity and initiating network connectivity complements Shell-Cole's prioritization of active applications. A POSITA would have recognized the benefit of integrating these teachings to

enhance resource allocation and connection management, ensuring that user-engaged applications receive timely access to network resources. The integration of Flack into Shell-Cole would have amounted to nothing more than the use of a known technique—Flack’s determination of application status based on user activity—to improve similar devices—Shell-Cole’s telephonic device—in a similar way—using connection policies—and combining these prior art teachings according to known, expressly disclosed methods to yield the predictable results described above. As applied here, incorporating Flack’s teachings into Shell-Cole provides a *processor* that is configured to determine whether a *user’s interaction* with an application (i.e., *a different application*) instigates connection requests, meaning that its *processor* determines that the application is an *active application* based on user input. Section XI.B.5-XI.B.6, XII.B; EX1007, ¶¶8, 16-19, 39, 50, 63. Therefore, claim 26 would have been obvious over Shell-Cole-Flack.

XIV. CONCLUSION

153. For the reasons set forth above, it is my opinion that claims 1 and 26 of the ’359 patent are unpatentable in view of the prior art. In signing this declaration, I understand that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I acknowledge that I may be subject to cross-examination in this case and that cross-examination will take place within the United States. If cross-examination is

required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

154. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: October 31, 2025

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'K Butler', written over a horizontal line.

Dr. Kevin Butler