

Appendix C – Combination References

Fortinet, Inc. ("Fortinet") provides this chart subject to all reservations, objections, statements, and disclaimers set forth herein and in Fortinet's Preliminary Invalidity Contentions Cover Pleading, as well as any amendment, supplement, or modification thereof, which are incorporated herein by reference in their entirety.

The following references (collectively the "Appendix References") are prior art to the '936 patent:

- Published PCT Application WO2001011452A2 to Wood, filed July 31, 2000, and published February 15, 2001, and thus prior art under at least 35 U.S.C. §102(a), (b), and (e).
- U.S. Patent Publication No. 2002/001077 to Lerner, filed December 28, 2000, and published January 24, 2002, and thus prior art under at least 35 U.S.C. §102(a) and (e).
- U.S. Patent No. 5,623,492 to Teraslinna, filed March 24, 1995, and issued April 22, 1997, and thus prior art under at least 35 U.S.C. §102(a), (b), and (e).
- U.S. Patent No. 7,197,044 to Kadambi, filed March 17, 2000, and issued March 27, 2007, and thus prior art under at least 35 U.S.C. §102(e).

As illustrated in the chart below, these references disclose or render obvious certain limitations or claims of the '639 patent and can be combined with each other or other charted references for the reasons set forth in the Cover Pleading. These Preliminary Invalidity Contentions are not an admission by Fortinet that the accused products, including any current or past versions of the accused products, are covered by, or infringe the asserted claims, but are based instead on the recognition that, if the claims are interpreted to be broad enough to encompass the accused products, the claims must also be construed to have that same scope when considering whether they are invalid. The fact that Fortinet has cited evidence for a preamble is not a concession that the preamble is limiting.

The following chart is partially based on, but is not limited by, the claim constructions implicit in Plaintiff's Infringement Contentions, to the extent that such constructions are apparent from the Infringement Contentions. Fortinet notes that in many instances, Plaintiff's Infringement Contentions fail to provide adequate notice of Plaintiff's construction of the asserted claims and fail to comply with the Court's scheduling order and other applicable rules. Fortinet does not accept the assumptions concerning the scope and meaning implicit in Plaintiff's Infringement Contentions, to the extent those assumptions are discernible, and reserves the right to challenge Plaintiff's proposed (or implied) constructions. Fortinet also reserves the right to revise and supplement these charts if and when Plaintiff is permitted to provide revised Infringement Contentions or otherwise make its positions known. To the extent that these Preliminary Invalidity Contentions rely on or otherwise embody particular constructions of terms or phrases in the asserted claims, Fortinet does not necessarily advocate any such construction as proper constructions of those terms or phrase. Fortinet also reserves the right to revise

and supplement these charts after the Court construes the claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Pincites in the chart below to Wood refer to page numbers in the complete document (including the title page, which is pincited as p. 1), rather than to the printed page numbers.

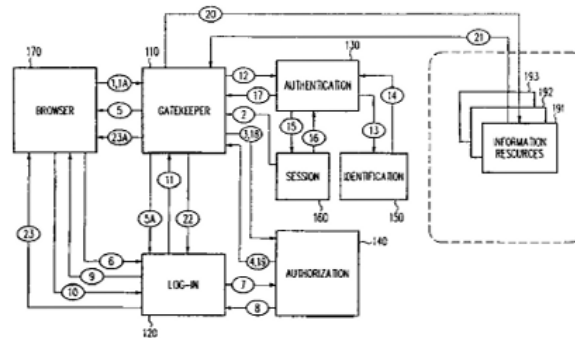
Claim Language	Exemplary Citations from Appendix References
<p>[1] A method for network access control, comprising:</p>	<p>The Appendix References disclose and/or render obvious: A method for network access control, comprising.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 3: Accordingly, a security architecture has been developed in which a single sign-on is provided. Session credentials are used to maintain continuity of a persistent session across multiple accesses to one or more information resources, and in some embodiments, across credential level changes. Session credentials are secured, e.g., as a cryptographically secured session token, such that they may be inspected by a wide variety of entities or applications to verify an authenticated trust level, yet may not be prepared or altered except by a trusted authentication service. Some embodiments of the present invention associate trust level requirements with information resources. Authentication schemes (e.g., those based on passwords, certificates, biometric techniques, smart cards, etc.) are associated with trust levels, and in some embodiments, with environmental parameters. For example, in one configuration, a login service obtains login credentials for an entity commensurate with the trust level requirement(s) of an information resource (or information resources) to be accessed and with environment parameters that affect the sufficiency of a given credential type. Once login credentials have been obtained for an entity and have been authenticated to a given trust level, session credentials are issued and access is granted to information resources for which the trust level is sufficient. Advantageously, by using the session credentials access is granted without the need for further login credentials and authentication. In some configurations, session credentials evidencing an insufficient trust level may be remedied by a session continuity preserving upgrade of login credential.</p> <p>Wood at p. 4: In still yet another embodiment in accordance with the present invention, an information access control facility includes an application proxy and means responsive to the application proxy. The application proxy is for receiving an access request targeting one of the information resources, extracting a cryptographically secured session token from the access request, and selectively proxying the access request. The means responsive to the application proxy is for evaluating</p>

	<p>sufficiency of an authorization encoded in the cryptographically secured session token for access to the targeted information resource.</p> <p>Wood at p. 4: FIG. 1 is a block diagram illustrating information flows between components in a security architecture employing secure session credentials in accordance with an exemplary embodiment of the present invention.</p> <p>Wood at p. 4: FIG. 2 is a flow chart illustrating operation of a security architecture employing secure session credentials in accordance with an exemplary embodiment of the present invention.</p> <p>Wood at p. 5: Access Management: Systems, methods and techniques for controlling use of information resources. Typically, access management systems employ both authentication and authorization to control access to information resources.</p> <p>Wood at p. 6: FIG. 1 provides an overview of major interactions between components for an exemplary security architecture in accordance with the present invention. As illustrated in FIG. 1, a client application, e.g., a browser 170 operated by a user, interacts with the security architecture via a gatekeeper and entry handler component 110 and a login component 120. Gatekeeper and entry handler component 110 provides an entry point for external client applications requesting access to enterprise applications and/or resources, including e.g., information resources 191, 192 193, for which access management is provided by the security architecture. Using facilities provided by a session management component 160, an authorization component 140, an authentication component 130, an identification component 150, and login component 120, the gatekeeper/entry handler component 110 allows, redirects or refuses access requests in accordance with a security policy.</p> <p>Wood at p. 7: In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 queries authorization component 140 to obtain authorization for access to a particular requested enterprise application or information resource by the requesting entity (e.g.,</p>
--	---

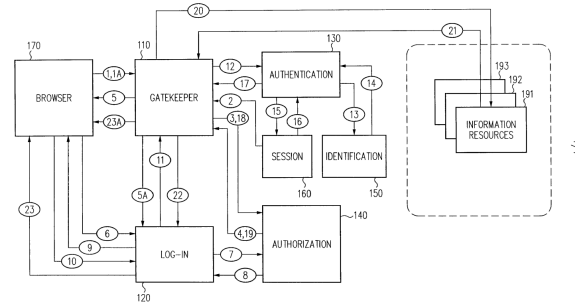
	<p>the browser user). If the entity requesting access has not yet been authenticated to the trust level required for the particular access to the particular enterprise application or information resource requested, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that login credentials may be obtained and authenticated to a particular trust level. If, on the other hand, login credentials have already been obtained for the requesting entity and the requesting entity has been authenticated using the obtained credentials such that the required trust level has been achieved, the access will typically be allowed without the need for further login credentials and authentication. In certain circumstances, authorization component 140 may indicate that the access is to be refused. For example, authorization component 140 may be programmed to perform more stringent testing beyond a trust level requirement. In an exemplary enterprise tool configuration, a desired security policy may dictate that a salary tool is accessible only from with a company's internal network. No level of authenticated trust may be sufficient to access such a tool from outside company network. To facilitate implementation of such a security policy, authorization component 140 could refuse access based on environment parameters indicating a session originating outside the company's internal network.</p> <p>Wood at p. 10: Referring to FIG. 1, an entity (e.g., a browser 170 operated by a user) interacts with enterprise applications and/or resources (e.g., 191, 192, 193) and the security architecture via a gatekeeper/entry handler component 110 and a login component 120. In general, a wide variety of entities, including human users operating browser and/or non-browser client applications as well as automated agents and systems, may interact with enterprise applications and/or resources and the security architecture as described herein. Furthermore, a variety of information resource identification schemes, such as by Uniform Resource Locator (URL), resource address, identifier or namespace designation, may be employed. However, for purposes of illustration and not limitation, an exemplary interaction involving a browser and information resources identified by URL is described in detail. Nonetheless, based on the description herein, persons of ordinary skill in the art will appreciate a wide variety of configurations in accordance with the present invention in which non-browser clients, automated agents or other systems interact with enterprise applications and/or resources and the security architecture using any of a variety of information resource identification schemes.</p>
--	---

	<p>Wood at p. 12-13:</p> <p>Gatekeeper/entry handler component 110 supplies authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session. Preferably, the associated session identifier is cryptographically secured (e.g., as a signed session credential). In some configurations, the signed session credential is obtained from the corresponding session object. In the case of a pre-existing session, the signed session credential may be obtained using a received session token. In any case, authorization component 140 receives (3) the requested resource and session identifiers and responds (4) in accordance with the trust level requirement of the requested resource. In configurations for which sensitivity to a changing environment is desired, environment information may also be supplied to authorization component 140. In an exemplary configuration, authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency of a current trust level. In some configurations, authorization component 140 dynamically calculates a current trust level based on the signed session credentials and environment information. In others, authorization component 140 may base its ALLOW, REDIRECT, or REFUSE response on a "current" trust level previously associated with the signed session credentials. Generally, an access request with sufficient current trust level is ALLOWED and forwarded (14) without further authentication. An authorization request without proper parameters (e.g., without a specified information resource or without a properly secured session credential) may be REFUSED. Authorization requests associated with a client entity that has been blacklisted or for a resource for which the associated client entity cannot be authenticated using any available method to achieve the required trust level may also be REFUSED. For example, a given security policy and associated trust level mappings may dictate a REFUSE response in response to an access request to a sensitive resource (such as financial data) by any client entity (even a browser supplying the digital certificate for the CFO, and therefore presumably operating on behalf of the CEO) if the access request is received over a dial-up line and originates from an unknown number or is received outside of working hours.</p>
--	---

Wood at Fig. 1:



Wood at Fig. 2:



Teraslinna '492 at 15:17-26:

Consider a customer who has a plurality of virtual connections provisioned from an endpoint of a fast packet switching network to a plurality of destination endpoints, wherein the fast packet switching network manages bandwidth resources in accordance with embodiments of the present invention. In order to avoid a discarding of packets by the network, it is beneficial for the customer to control the flow of packets being forwarded to the endpoint. Embodiments of methods and systems for controlling the flow of packets presented to the network are described next.

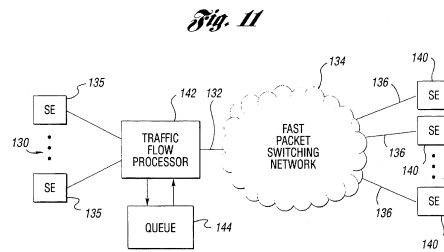
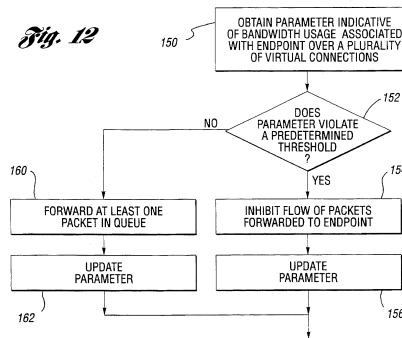
Teraslinna '492 at 15:27-39:

FIG. 11 is a block diagram of an embodiment of a system for controlling a flow of packets being forwarded by customer premises equipment (CPE), generally indicated by reference

	<p>numeral 130, to an endpoint 132 of a fast packet switching network 134. The customer premises equipment 130 includes one or more types of subscriber equipment 135 in communication with the network 134. The fast packet switching network 134 has a plurality of virtual connections provisioned from the endpoint 132 to a plurality of destination endpoints 136. The virtual connections are utilized by the customer for communicating packets from the customer premises equipment 130 to subscriber equipment 140 at one or more of the destination endpoints.</p> <p>Teraslinna '492 at 15:40-49: The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint 132 over at least two of the virtual connections. The at least one parameter indicative of bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.</p> <p>Teraslinna '492 at 15:50-59: The at least one parameter parallels at least one of the bandwidth parameters monitored and enforced by the network 134 in accordance with a predetermined traffic contract. Preferably, the traffic flow processor 142 obtains each of the parameters which are monitored by the network 134 in accordance with the predetermined traffic contract. The at least one parameter may be indicative of an average bit rate or a peak bit rate, for example, associated with the bundle. The at least one parameter may include a leaky bucket parameter as described earlier.</p> <p>Teraslinna '492 at 15:60-67: The traffic flow processor 142 further provides means for inhibiting the flow of packets forwarded to the endpoint 132 when the at least one parameter violates the traffic contract. Typically, this occurs when one of the at least one parameter violates a predetermined threshold. The predetermined threshold may be violated if the parameter exceeds the threshold, or alternatively, if the parameter drops below the threshold.</p> <p>Teraslinna '492 at 16:13-21: When the at least one parameter complies with the predetermined traffic contract, i.e., each parameter complies with a corresponding predetermined threshold, the traffic flow processor</p>
--	--

	<p>142 performs a step of forwarding one of the at least one packet stored in the queue 144 to the endpoint 132. The traffic flow processor 142 further updates each of the at least one parameter when the packet is forwarded to the endpoint 132. Typically, the leaky bucket parameter is updated by performing a step of incrementing the parameter.</p> <p>Teraslinna '492 at 16:42-51: As indicated by block 150, the method includes a step of obtaining a parameter indicative of bandwidth usage associated with the endpoint over at least two of the virtual connections. The parameter provides a measure of bandwidth usage associated with a bundle of provisioned virtual connections which include the at least two of the virtual connections. As previously mentioned, the parameter may be a measure of average bit rate or peak bit rate, and may be computed using a leaky bucket algorithm.</p> <p>Teraslinna '492 at 16:51-58: If the parameter violates a predetermined threshold specified by a predetermined traffic contract, as determined by block 152, a step of inhibiting the flow of packets being forwarded to the endpoint is performed, as indicated by block 154. Consequently, when the flow is inhibited, the number of packets stored in the queue increases as further packets are provided by the customer premises equipment.</p>
--	---

Teraslinna '492 at Fig. 11-12:



Lerner '776 at [0014]:

In accordance with one embodiment of the present invention, there is provided a method of providing a single sign-on distributed application services integration, comprising the steps of: receiving a first indication of a user pointing a browser to a first application; receiving a cookie file of said browser corresponding to the user; updating said cookie file; receiving a second indication of said user pointing said browser to a second application; and providing said updated cookie file to said second application.

Lerner '776 at [0015]:

In accordance with another embodiment of the present invention, there is provided a system for providing a single sign-on distributed application services integration, comprising: a client terminal; and a central server coupled to said client terminal configured to receive from said

	<p>client terminal a first indication of a user pointing a browser to a first application and a cookie file of said browser corresponding to the user; wherein said central server is further configured to update said cookie file, and when a second indication of said user pointing said browser to a second application is received from said client terminal, said central server provides said updated cookie file to said second application.</p> <p>Lerner '776 at [0038]: In one aspect of the present invention, a client or a user at a client terminal points the client web browser 270 at a particular resource (page, CGI, script, etc) that is part of the web application and which generates a need to make information about that event available to other web applications in real time. The cookie files for the server's central domain are then automatically passed from the client web browser 270 to the web application 210 . The web application 210 then calls an application interface library function from the application interface library 220 to make the needed information available. The type of information to be shared and the data specific to this particular event or instance is then passed to the function as a parameter in the form of one or more NAME=VALUE pairs. The function calls cookie access library functions from the cookie access library 240 to read the current data in the cookie. Moreover, if necessary, the cookie is encrypted by the encryption/ decryption engine 230.</p> <p>Lerner '776 at [0039]: The cookie access library 240 then writes the data in the cookie file back to the client web browser 270 via the internet 250 . The client subsequently points the client web browser 270 to a resource that is part of a different web application on a different site but still within the central server's domain that needs the information generated by the client's activity in the previous web application. The new web application then calls the application interface library function from the application interface library 220 to request the desired information using the name part of the NAME=VALUE pair used to store the data initially. The application interface library function then calls cookie access library functions from the cookie access library 240 to read the data in the cookie, and if the cookie to be read is encrypted, the cookie is first decrypted by the encryption/ decryption engine 230 . The application is now configured to perform specific action based on the value returned.</p>
--	--

	Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.
[1.1] at a control device, receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user;	<p>The Appendix References disclose and/or render obvious: at a control device, receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 2: The internet has become an important medium for information services and electronic commerce. As the internet has been commercialized, organizations initially established their presence in cyberspace by making information (typically static, non-sensitive promotional information) available on resources well removed from the operational infrastructure of the organization. Security issues were often addressed by isolating publicly accessible resources (e.g., web servers) from more sensitive assets using firewall techniques. As long as the publicly accessible information and resources were relatively non-sensitive and user interactions with such information and resources was not mission critical, relatively simple firewall techniques were adequate. Though information and resources outside the firewall were at risk, the risk could generally be limited to non-proprietary information that was easily replaceable if compromised. Proprietary information and systems critical to day-to-day operations were sheltered behind the firewall and information flows across the firewall were filtered to exclude all but the comparatively non-threatening services such as electronic mail.</p> <p>Wood at p. 4: FIG. 2 is a flow chart illustrating operation of a security architecture employing secure session credentials in accordance with an exemplary embodiment of the present invention.</p> <p>Wood at p. 5: Entity: A user or object, including data objects and/or functional objects such as applications, components, modules, services, processes, threads, etc., as well as instantiations thereof. In some configurations, only user entities (typically, the human principal interacting with a software program or on whose behalf a software agent purports to act) are considered. In other configurations, entities include functional objects without an associated human principal. The identity of an entity may be authenticated using a credential.</p>

Wood at p. 6:

FIG. 1 provides an overview of major interactions between components for an exemplary security architecture in accordance with the present invention. As illustrated in FIG. 1, a client application, e.g., a browser 170 operated by a user, interacts with the security architecture via a gatekeeper and entry handler component 110 and a login component 120. Gatekeeper and entry handler component 110 provides an entry point for external client applications requesting access to enterprise applications and/or resources, including e.g., information resources 191, 192 193, for which access management is provided by the security architecture. Using facilities provided by a session management component 160, an authorization component 140, an authentication component 130, an identification component 150, and login component 120, the gatekeeper/entry handler component 110 allows, redirects or refuses access requests in accordance with a security policy.

Wood at p. 7:

In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 queries authorization component 140 to obtain authorization for access to a particular requested enterprise application or information resource by the requesting entity (e.g., the browser user). If the entity requesting access has not yet been authenticated to the trust level required for the particular access to the particular enterprise application or information resource requested, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that login credentials may be obtained and authenticated to a particular trust level. If, on the other hand, login credentials have already been obtained for the requesting entity and the requesting entity has been authenticated using the obtained credentials such that the required trust level has been achieved, the access will typically be allowed without the need for further login credentials and authentication. In certain circumstances, authorization component 140 may indicate that the access is to be refused. For example, authorization component 140 may be programmed to perform more stringent testing beyond a trust level requirement. In an exemplary enterprise tool configuration, a desired security policy may dictate that a salary tool is accessible only from with a company's internal network. No level of authenticated trust may be sufficient to access such a tool from outside company network. To facilitate implementation of such a security policy, authorization component 140 could refuse

	<p>access based on environment parameters indicating a session originating outside the company's internal network.</p> <p>Wood at p. 10-11: Focusing then on an exemplary browser-type client entity, browser 170 requests access to one or more of enterprise applications and/or resources (e.g., information resource 191) by presenting an URL to gatekeeper/entry handler component 110, which acts as a point of entry for client entities requesting access to applications and/or resources controlled by the security architecture. Gatekeeper/entry handler component 110 receives the request as a proxy for the requested resource. In some configurations, a combined gatekeeper/entry handler instance is provided, while in others, separate and/or multiple instances are provided with functionally identical interfaces to other components of the security architecture. In some configurations, multiple instances of entry handler functionality (e.g., interception of inbound requests and collection of environment information) are provided. For example, one or more instances for each of several connection types (e.g., dialup, WAN, etc.) may be employed. One or more instances of gatekeeper functionality (e.g., allowing access for authorized requests and otherwise denying or initiating appropriate responsive action) may also be provided. Configurations and functional decompositions suitable to a particular environment and expected load requirements will be appreciated by persons of ordinary skill in the art.</p> <p>Wood at p. 11: Gatekeeper functionality (e.g., in gatekeeper/entry handler component 110) checks whether a session is already associated with the incoming request. Although other techniques are possible, in some configurations in accordance with the present invention, gatekeeper/entry handler component 110 checks for the presence of a session token in the incoming request. Use of session tokens is described in greater detail below; however, in short, a session token may be any data supplied to the client entity for use in uniquely identifying an associated session. In general, preferred session token implementations are cryptographically secured and include facilities, such as expiration or mapping to a particular connection, to limit risk of replay attack and/or spoofing. Some session token implementations may encode session, principal, and/or trust level information. Some session token implementations may employ cookies, URL encoding, or other similar techniques for binding to incoming requests.</p>
--	---

	<p>Wood at p. 15-16:</p> <p>Login component 120 also passes control of the access request back to gatekeeper/entry handler component 110. In an exemplary configuration, login component 120 issues a new HTTP request (11) specifying the originally requested URL, which has been stored in the HttpSession object. As before, gatekeeper/entry handler component 110 receives the request. Gatekeeper/entry handler component 110 extracts the login credentials from the request or from the HttpSession object and passes (12) the login credentials to authentication component 130 for authentication. Authentication component 130 authenticates the login credential, and if successful, queries (13) identification component 150 to establish a correspondence with a set of entity descriptors that uniquely identifies the requesting entity. In an exemplary configuration, entity descriptor types include: entity id, system id (e.g., name/password), certificate, enigma id, smartcard token, name/address, and phone. One or more of values may uniquely identify an entity and one or more entity descriptor types may support multiple values (e.g., multiple digital certificates, name/password pairs, or phone numbers per identity). Once the requesting entity has been identified (14), session parameters are updated (15) and session management component 160 supplies (16) session credentials. Preferably, session credentials are digitally-signed or otherwise cryptographically secured and returned (17) to gatekeeper/entry handler component 110.</p> <p>Wood at p. 16-17:</p> <p>Referring again to FIG. 1, subsequent access requests (e.g., access request 1A) include a previously assigned session token. As described above, gatekeeper/entry handler component 110 uses the session token, if provided to resolve a session object containing session credentials, and to determine whether previously authenticated credentials are sufficient for the requested access. As described above, authorization component 140 may be queried using session credentials and an identifier for the requested resource to determine sufficiency of previously authenticated credentials. In some configurations, sufficiency is determined using trust level mappings as described above. Depending on the information resource to which access is requested, and in some configurations depending on current session environment information, access request 1A may or may not have associated previously authenticated credentials sufficient to support the requested access. In the case of an access request 1A having a trust level requirement commensurate with previously obtained and authenticated credentials (i.e., an access request for which no additional credentials need be obtained via login component 120), the access request is proxied (20) and results (21) are streamed directly (23A) back to browser 170. In some</p>
--	--

configurations, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated session token, if supplied, resolves to the same session object as the session token replaced. For example, in some configurations, both session tokens encode a same session handle, although other aspects associated with session token security such as expiration may be updated. In other configurations, results (21) are streamed (23A) back to browser 170 without an updated session token. In such configurations, the previously supplied session token remains valid until expired or otherwise invalidated. Some variations may employ a session token refresh interval.

Wood at p. 18:

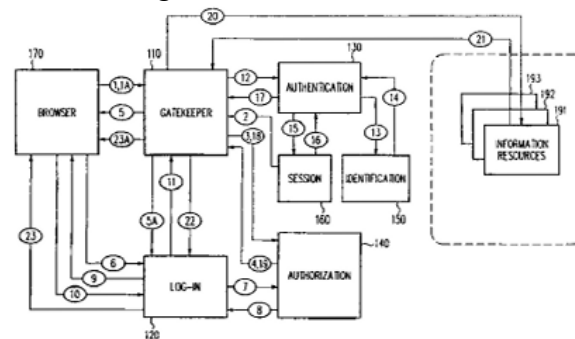
FIG. 2 illustrates operation of an exemplary security architecture providing a single sign-on facility with trust level mapping to authentication requirements. As before, an access request is received (201) from a client entity. If the request does not contain a session identifier (e.g., a session key or token) or if the request can otherwise be reliably associated with a session maintained by the security architecture, a new session is created (202). A trust level requirement is determined for access to the requested resource in the context of the requesting session environment. In some configurations, as described above, the determination is performed by an authorization service such as authorization component 140. Given a trust level requirement, current session credentials are evaluated (203) in the context of session environment information to determine whether the previously supplied login credentials are sufficient to achieve the required trust level. In one advantageous realization of session credentials and tokens, a cryptographically secured encoding of trust level allows authorization to be confirmed without involvement of an authentication service (e.g., with reauthentication of login credentials). In the case of a newly created (202) session, the authorization evidenced by session credentials will typically be insufficient, although some security policies may allow anonymous access to certain resources.

Wood at p. 19:

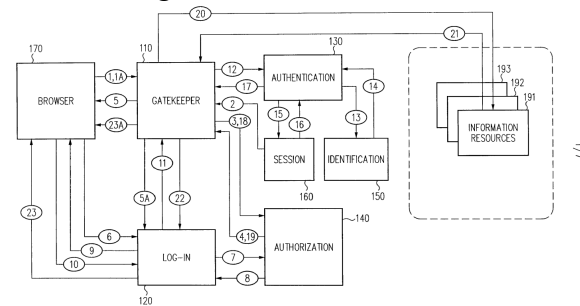
Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others

the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information. Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents back to the credential gathering service. Others (e.g., certificates) may be obtained for the client entity from an agent or authority. For example, a personal digital certificate (such as those issued by VerisignTM, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.

Wood at Fig. 1:



Wood at Fig. 2:



Teraslinna '492 at Abstract:

Methods and systems for resource management in a fast packet switching communication network are disclosed. A packet containing an address field identifying a source endpoint and a destination endpoint is received. A bandwidth constraint based upon the source endpoint, but independent of the destination endpoint, is enforced for the packet. In a communication network wherein a plurality of virtual connections are provisioned from a single source endpoint to a plurality of destination endpoints, the bandwidth constraint is associated with bandwidth usage over all of the virtual connections.

Teraslinna '492 at 1:9-13:

The present invention relates generally to managing bandwidth resources within a switching communication network, and more particularly, to managing bandwidth resources within a switching communication network having at least one fast packet switch.

Teraslinna '492 at 3:25-26:

FIG. 5 is a block diagram of a communication network in accordance with embodiments of the present invention;

Teraslinna '492 at 4:1-10:

FIG. 1 is a flow chart of an embodiment of a method of resource management in a fast packet switching network. The fast packet switching network, having a plurality of endpoints, includes at least one fast packet switch used in providing a virtual connection between at least

	<p>one selected pair of the endpoints. The method includes a step of receiving a packet having an address field which identifies a source endpoint and a destination endpoint, as indicated by block 10. Typically, the step of receiving the packet is performed at a switch port in a fast packet switch within the network.</p> <p>Teraslinna '492 at 6:3-15: FIG. 3 is a block diagram of a fast packet switch 30 having an embodiment of a system for bandwidth resource management in accordance with the present invention. The fast packet switch 30 includes a plurality of switch ports 32, each of the switch ports 32 providing a port either to some type of subscriber equipment (for communication with the subscriber) or to a transmission facility (for communication with another switch in a network). It is noted that subscriber equipment refers generally to any type of equipment which a subscriber may employ at an endpoint of the network. Examples of subscriber equipment include, but are not limited to, phone equipment, a router or other datacom equipment, a private branch exchange (PBX), or an interface to another network.</p> <p>Teraslinna '492 at 6:24-33: A packet protocol processor 37 performs the packet protocol processing. The packet protocol processor 37 provides means for enforcing a bandwidth constraint based upon an address field contained within the received packet. Consequently, the line unit 34 enforces a bandwidth constraint for packets communicated over a bundle of provisioned virtual connections emanating from a source endpoint identified by the address field. As described earlier, a received packet which violates the bandwidth constraint is penalized in a predetermined manner.</p> <p>Teraslinna '492 at 7:33-42: In order to illustrate the utilization of embodiments of the present invention and advantages which result therefrom, a block diagram of an example communication network in accordance with the present invention is given in FIG. 5. For the purpose of illustration, the network includes three representative fast packet switches 70, 72, and 74. The fast packet switch 70 communicates with the fast packet switch 72 via a transmission facility 76. Similarly, the fast packet switch 72 communicates with the fast packet switch 74 via a transmission facility 80.</p>
--	---

Teraslinna '492 at 15:27-39:

FIG. 11 is a block diagram of an embodiment of a system for controlling a flow of packets being forwarded by customer premises equipment (CPE), generally indicated by reference numeral 130, to an endpoint 132 of a fast packet switching network 134. The customer premises equipment 130 includes one or more types of subscriber equipment 135 in communication with the network 134. The fast packet switching network 134 has a plurality of virtual connections provisioned from the endpoint 132 to a plurality of destination endpoints 136. The virtual connections are utilized by the customer for communicating packets from the customer premises equipment 130 to subscriber equipment 140 at one or more of the destination endpoints.

Teraslinna '492 at Fig. 5:

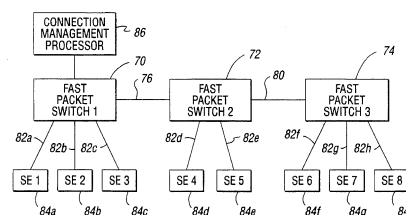
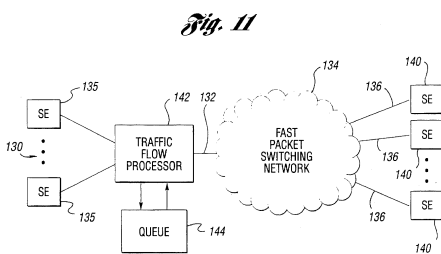
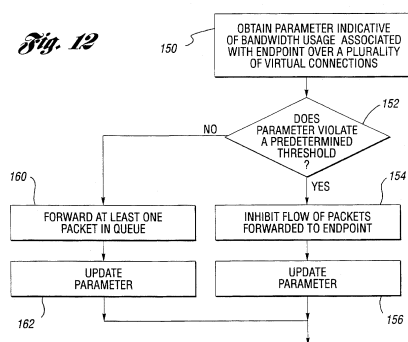


Fig. 5

Teraslinna '492 at Fig. 11-12:



Teraslinna '492 at Fig. 2a-2b:

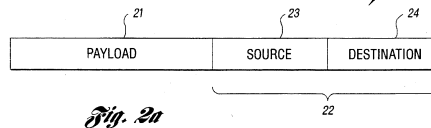


Fig. 2a

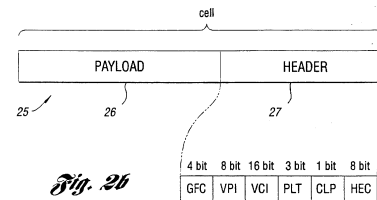
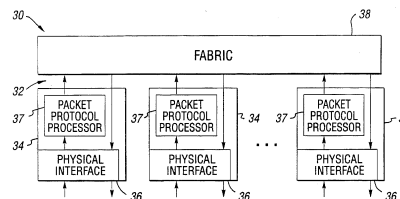


Fig. 2b

Teraslinna '492 at Fig. 3:



Lerner '776 at [0049]:

The message broker 390 at the central server receives the message transmitted from the web application 310 and determines, based on the content of the message, which other applications or data stores in the central server domain need to receive it. The message broker 390 further performs any necessary data translation such that the message is compatible to those applications or data stores that are to receive the message, and thus make of the data contained in the transmitted message. Alternatively, the message broker 390 may directly communicate with local applications or data stores residing in the central server terminal, or may encrypt and queue messages as needed to transmit the messages to remote locations over a public data network such as the internet.

	<p>Lerner '776 at [0063]:</p> <p>First, at step 501 the user terminal client web browser 412 submits an HTTP request with a cookie to a web application 401 such that the URL of the desired resource (e.g., application 401) available in the central server domain and the cookie is transmitted to the application 401 . Having received the HTTP request and the cookie, at step 502 the application corresponding to the HTTP request (e.g., application 401) transmits the cookie to the application interface library 402 . The application interface library 402 at step 503 then performs user authentication processing. In particular, the application interface library 402 transmits the cookie received from the client web browser 412 to the cookie access library 405 which is configured to read the cookie data at step 504 . In the event that the transmitted cookie is encrypted, the cookie access library 405 transmits the encrypted cookie to the encryption/ decryption engine 403 which, at step 505 decrypts the encrypted cookie and passes the decrypted cookie back to the cookie access library 405 . The cookie access library 405 then parses the decrypted cookie at step 506 into cookie data and accordingly, determines whether the user ID is matched or not by returning a confirm authentication or a null match command to the application interface library 402 . With a matched user ID, the application interface library 402 returns the result to the web application 401 which then delivers the application content at step 507 for display at the client browser 412 at step 508 . On the other hand, if the authentication process is unsuccessful, a null match command is returned to the web application 401 which then delivers redirect information with a HTTP redirect to the client web browser 412 at step 509 . The client web browser 412 then submits the HTTP redirect request to the login page of the application at step 510 , and the login page URL and the cookie is sent to the login page at the client web browser 412 at step 511 where the user is prompted to input login information to access the desired application.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[1.2] processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage;	<p>The Appendix References disclose and/or render obvious: processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 3:</p>

	<p>Accordingly, a security architecture has been developed in which a single sign-on is provided. Session credentials are used to maintain continuity of a persistent session across multiple accesses to one or more information resources, and in some embodiments, across credential level changes. Session credentials are secured, e.g., as a cryptographically secured session token, such that they may be inspected by a wide variety of entities or applications to verify an authenticated trust level, yet may not be prepared or altered except by a trusted authentication service. Some embodiments of the present invention associate trust level requirements with information resources. Authentication schemes (e.g., those based on passwords, certificates, biometric techniques, smart cards, etc.) are associated with trust levels, and in some embodiments, with environmental parameters. For example, in one configuration, a login service obtains login credentials for an entity commensurate with the trust level requirement(s) of an information resource (or information resources) to be accessed and with environment parameters that affect the sufficiency of a given credential type. Once login credentials have been obtained for an entity and have been authenticated to a given trust level, session credentials are issued and access is granted to information resources for which the trust level is sufficient. Advantageously, by using the session credentials access is granted without the need for further login credentials and authentication. In some configurations, session credentials evidencing an insufficient trust level may be remedied by a session continuity preserving upgrade of login credential.</p> <p>Wood at p. 5: Entity: A user or object, including data objects and/or functional objects such as applications, components, modules, services, processes, threads, etc., as well as instantiations thereof. In some configurations, only user entities (typically, the human principal interacting with a software program or on whose behalf a software agent purports to act) are considered. In other configurations, entities include functional objects without an associated human principal. The identity of an entity may be authenticated using a credential.</p> <p>Wood at p. 6: FIG. 1 provides an overview of major interactions between components for an exemplary security architecture in accordance with the present invention. As illustrated in FIG. 1, a client application, e.g., a browser 170 operated by a user, interacts with the security architecture via a gatekeeper and entry handler component 110 and a login component 120. Gatekeeper and entry handler component 110 provides an entry point for external client applications requesting access</p>
--	--

	<p>to enterprise applications and/or resources, including e.g., information resources 191, 192 193, for which access management is provided by the security architecture. Using facilities provided by a session management component 160, an authorization component 140, an authentication component 130, an identification component 150, and login component 120, the gatekeeper/entry handler component 110 allows, redirects or refuses access requests in accordance with a security policy.</p> <p>Wood at p. 6: Individual information resources typically have differing security requirements. In addition, individual types of access to a single information resource may have differing security requirements. Nonetheless, a given level of security may be sufficient for more than one of the information services or access types. For example, information resource 191 may include a product information service for providing general information such as product descriptions or data sheets to the public, while information resource 192 includes an order processing system for an eCommerce site. Information resource 193 may include functions for supply chain interactions such as access to inventory information or current selling price information. Both the product information service and order intake functions of the eCommerce may operate with similar security requirements, e.g., allowing access by minimally authenticated, non-hostile entities. On the other hand, supply chain functions may require a higher level of security. Order status functions of the order processing system may require a mid-level of security.</p> <p>Wood at p. 7: Of note, in certain embodiments in accordance with the present invention, the mapping of login credential types and authentication mechanisms to trust levels is influenced by environment information such as time of request, source of request, connection speed, and/or client application (e.g., browser) environment information. In this way, even with a static set of mapping rules, the set of credential types and authentication mechanisms suitable to support a given trust level may vary based on environment information. In general, mapping rule dependencies are based on perceived variations in threat characteristics and/or requesting entity capabilities. In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 is the authority on environment information for a particular requesting entity.</p>
--	--

	<p>Wood at p. 7:</p> <p>In some embodiments, mapping rules may be dynamically varied. For example, if a particular login credential type and/or authentication method is deemed insecure (e.g., because compromised or because of a changing threat profile), the trust level mappings can be updated and have enterprise-wide effect. In addition, several other advantages are achieved by defining the authentication requirement interface between enterprise applications and/or resources and the security architecture in terms of required trust levels, rather than in terms of particular credential types and authentication methods. First, single sign-on configurations are facilitated using an enterprise-wide credential obtaining, authentication and session tracking infrastructure. Second, authentication requirements may be enforced uniformly in accordance with an enterprise-wide security policy and with reduced vulnerability to a lax security implementation by any particular information resource. Third, credential types and authentication methods can be added, deleted, or mapped to a new trust level, all without modification of enterprise applications and resources. Of course, all advantages need not be achieved in any particular implementation.</p> <p>Wood at p. 8:</p> <p>In addition to the obtained login credentials, some configurations in accordance with the present invention employ session credentials as a mechanism for evidencing prior authentication of obtained login credentials and for binding individual transactions to a particular session. In some configurations, session credentials are also employed in a session token form advantageous for transactions external to the security architecture. In an exemplary realization, session tokens are encoded for supply to browsers as cookies. FIG. 4 illustrates relationships between exemplary login credential, session credential and session token objects.</p> <p>Wood at p. 9:</p> <p>Referring again to session credentials structure 420, a session id, a principal id, a trust level, group ids, a creation time and an expiration time are encoded in both in encrypted portion 421 and clear text portion 422. The session id is a unique identifier for a persistent session maintained by the security architecture. In implementations in which credential upgrade is provided or in which a session credential expiration and refresh is provided, multiple successively issued session credential instances may encode the same session id and correspond to the same persistent session. Principal id encodes an identifier for a principal to which the security</p>
--	---

	<p>architecture has resolved login credentials. For example, a login credential including a username jdoe and a password corresponding to jdoe may be resolved by the security architecture to a unique principal id associated with John. Q. Doe of the shipping and receiving department, having an employee badge number of 12345, etc.</p> <p>Wood at p. 9:</p> <p>In some embodiments, a trust level encodes the authorization level to which a principal has been authenticated. In such embodiments, the encoded trust level serves as a basis for evaluating whether a principal associated with the session credentials has been authenticated to a sufficient level for access to a requested resource. For example, a trust level of 5 may be sufficient for access to information resources having a trust level requirement of 5 or less. Trust levels offer an attractive decoupling of authorization levels and authentication methods as described elsewhere herein. However, in some embodiments, an authorization encoding may establish that a principal has been authenticated using a particular authentication mechanism. In either case, an authorization (e.g., encoded as a trust level) in a cryptographically secured session credential allows the security architecture to authorize accesses based on prior authentication of a login credential and without involvement of the authentication service.</p> <p>Wood at p. 10-11:</p> <p>Focusing then on an exemplary browser-type client entity, browser 170 requests access to one or more of enterprise applications and/or resources (e.g., information resource 191) by presenting an URL to gatekeeper/entry handler component 110, which acts as a point of entry for client entities requesting access to applications and/or resources controlled by the security architecture. Gatekeeper/entry handler component 110 receives the request as a proxy for the requested resource. In some configurations, a combined gatekeeper/entry handler instance is provided, while in others, separate and/or multiple instances are provided with functionally identical interfaces to other components of the security architecture. In some configurations, multiple instances of entry handler functionality (e.g., interception of inbound requests and collection of environment information) are provided. For example, one or more instances for each of several connection types (e.g., dialup, WAN, etc.) may be employed. One or more instances of gatekeeper functionality (e.g., allowing access for authorized requests and otherwise denying or initiating appropriate responsive action) may also be provided. Configurations and functional</p>
--	---

	<p>decompositions suitable to a particular environment and expected load requirements will be appreciated by persons of ordinary skill in the art.</p> <p>Wood at p. 11: Gatekeeper functionality (e.g., in gatekeeper/entry handler component 110) checks whether a session is already associated with the incoming request. Although other techniques are possible, in some configurations in accordance with the present invention, gatekeeper/entry handler component 110 checks for the presence of a session token in the incoming request. Use of session tokens is described in greater detail below; however, in short, a session token may be any data supplied to the client entity for use in uniquely identifying an associated session. In general, preferred session token implementations are cryptographically secured and include facilities, such as expiration or mapping to a particular connection, to limit risk of replay attack and/or spoofing. Some session token implementations may encode session, principal, and/or trust level information. Some session token implementations may employ cookies, URL encoding, or other similar techniques for binding to incoming requests.</p> <p>Wood at p. 14: Generally, mapping rule logic is typically evaluated before a user is challenged to authenticate. Mapping occurs as a function of session environment and particulars of the information resource for which access is requested. By evaluating the minimum trust level required by the target of an access request, a service (e.g., a login service such as provided by login component 120) derives a list of potential authentication methods. The service then checks current session environment against the allowed environment states for each potential authentication method to trim the list further. If there is no particular resource for which access is being requested (e.g., if a user jumps straight to a sign-on page without requesting an access), the service will proceed according to the lowest level of trust available consistent with session environment. Other configurations may employ differing default behaviors.</p> <p>Wood at p. 15-16: Login component 120 also passes control of the access request back to gatekeeper/entry handler component 110. In an exemplary configuration, login component 120 issues a new HTTP request (11) specifying the originally requested URL, which has been stored in the HttpSession object. As before, gatekeeper/entry handler component 110 receives the request. Gatekeeper/entry handler component 110 extracts the login credentials from the request or from the HttpSession</p>
--	---

object and passes (12) the login credentials to authentication component 130 for authentication. Authentication component 130 authenticates the login credential, and if successful, queries (13) identification component 150 to establish a correspondence with a set of entity descriptors that uniquely identifies the requesting entity. In an exemplary configuration, entity descriptor types include: entity id, system id (e.g., name/password), certificate, enigma id, smartcard token, name/address, and phone. One or more of values may uniquely identify an entity and one or more entity descriptor types may support multiple values (e.g., multiple digital certificates, name/password pairs, or phone numbers per identity). Once the requesting entity has been identified (14), session parameters are updated (15) and session management component 160 supplies (16) session credentials. Preferably, session credentials are digitally-signed or otherwise cryptographically secured and returned (17) to gatekeeper/entry handler component 110.

Wood at p. 16:

Gatekeeper/entry handler component 110 again supplies (18) authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session (e.g., the signed session credentials. authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency the session credentials (and underlying authentication of login credentials) for the trust level required for the requested access. Login credentials should now be sufficient for access to the requested resource and an ALLOW response is supplied (19) by authorization component 140. Gatekeeper/entry handler component 110 proxies the requested access (20, 21) to information resource 191 and streams (22) results back to login component 120. Login component 120, in turn, streams (23) results back to browser 170.

Wood at p. 16-17:

Referring again to FIG. 1, subsequent access requests (e.g., access request 1A) include a previously assigned session token. As described above, gatekeeper/entry handler component 110 uses the session token, if provided to resolve a session object containing session credentials, and to determine whether previously authenticated credentials are sufficient for the requested access. As described above, authorization component 140 may be queried using session credentials and an identifier for the requested resource to determine sufficiency of previously authenticated credentials. In some configurations, sufficiency is determined using trust level mappings as described above. Depending on the information resource to which access is requested, and in

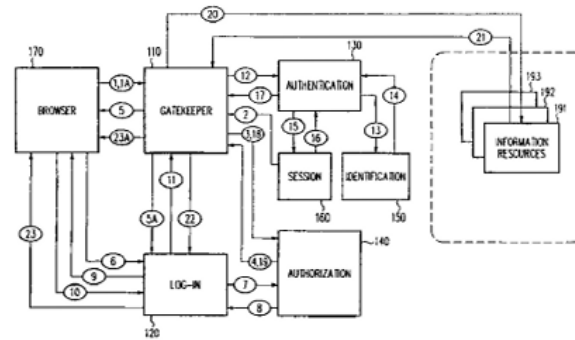
some configurations depending on current session environment information, access request 1A may or may not have associated previously authenticated credentials sufficient to support the requested access. In the case of an access request 1A having a trust level requirement commensurate with previously obtained and authenticated credentials (i.e., an access request for which no additional credentials need be obtained via login component 120), the access request is proxied (20) and results (21) are streamed directly (23A) back to browser 170. In some configurations, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated session token, if supplied, resolves to the same session object as the session token replaced. For example, in some configurations, both session tokens encode a same session handle, although other aspects associated with session token security such as expiration may be updated. In other configurations, results (21) are streamed (23A) back to browser 170 without an updated session token. In such configurations, the previously supplied session token remains valid until expired or otherwise invalidated. Some variations may employ a session token refresh interval.

Wood at p. 18:

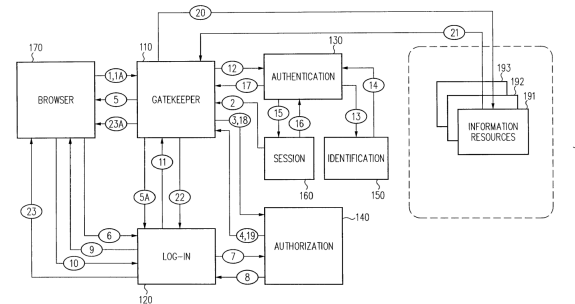
FIG. 2 illustrates operation of an exemplary security architecture providing a single sign-on facility with trust level mapping to authentication requirements. As before, an access request is received (201) from a client entity. If the request does not contain a session identifier (e.g., a session key or token) or if the request can otherwise be reliably associated with a session maintained by the security architecture, a new session is created (202). A trust level requirement is determined for access to the requested resource in the context of the requesting session environment. In some configurations, as described above, the determination is performed by an authorization service such as authorization component 140. Given a trust level requirement, current session credentials are evaluated (203) in the context of session environment information to determine whether the previously supplied login credentials are sufficient to achieve the required trust level. In one advantageous realization of session credentials and tokens, a cryptographically secured encoding of trust level allows authorization to be confirmed without involvement of an authentication service (e.g., with reauthentication of login credentials). In the case of a newly created (202) session, the authorization evidenced by session credentials will typically be insufficient, although some security policies may allow anonymous access to certain resources.

	<p>Wood at p. 19:</p> <p>Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information. Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents back to the credential gathering service. Others (e.g., certificates) may be obtained for the client entity from an agent or authority. For example, a personal digital certificate (such as those issued by Verisign™, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.</p>
--	--

Wood at Fig. 1:



Wood at Fig. 2:



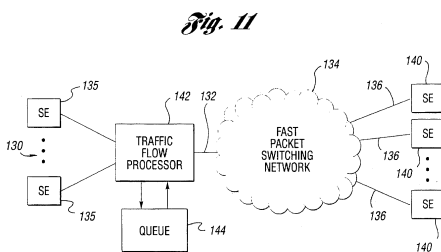
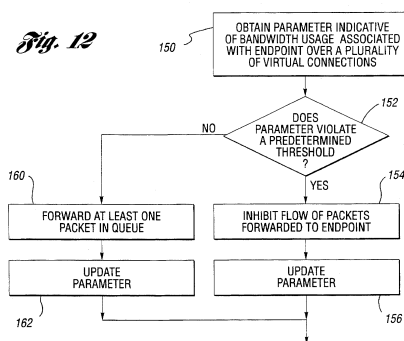
Teraslinna '492 at 1:35-47:

Each message to be communicated using fast packet switching is packetized by dividing the message into one or more pieces of information. Each of the pieces of information is loaded into the payload field of a corresponding packet. In a frame relay service, the packets are referred to as frames, and the length of the payload field is variable. In contrast, the length of the payload field is fixed in a cell relay service, wherein the packets are referred to as cells. Regardless of the type of packet employed, each of the packets is further loaded with an address field which contains information identifying a provisioned virtual connection between the two endpoints. Typically, the address field is contained in a header in the packet.

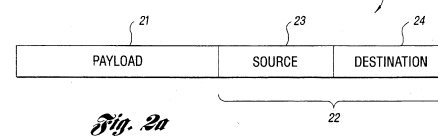
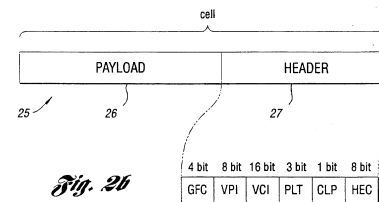
	<p>Teraslinna '492 at 5:1-10: Referring back to FIG. 1, the method further includes a step of enforcing a bandwidth constraint, as indicated by block 12, based upon the source endpoint identified by the address field, wherein the bandwidth constraint is independent of the destination endpoint identified by the address field. In a preferred embodiment wherein the address field contains a source label which identifies the source endpoint and a destination label which identifies the destination endpoint, the bandwidth constraint is enforced based upon the source label and is independent of the destination label.</p> <p>Teraslinna '492 at 6:45-49: FIG. 4 is a flow chart of a method of processing a received packet for use with embodiments of the line unit in accordance with the present invention. Upon receiving the packet, the address field, typically located in a header in the packet, is extracted as indicated by block 40.</p> <p>Teraslinna '492 at 8:10-25: Embodiments of the present invention are further illustrated in terms of a virtual private network, which is to be implemented for a plurality of subscribers. First, the network is provisioned with virtual connections between all subscriber endpoints. This consists of writing data into all the fast packet switches associating addresses and bandwidth parameters with all the virtual connections. This is performed by the network managers. When a transmission of information is to be made from a first subscriber to a second subscriber, the first subscriber packetizes the information, and includes the proper address field in a header of each packet. For two-way communications, the second subscriber packetizes its information to be sent to the first subscriber, and includes the proper address field in the header of each packet. The proper address field can be found by the subscriber in the following two ways.</p> <p>Teraslinna '492 at 15:40-49: The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint 132 over at least two of the virtual connections. The at least one parameter indicative of bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.</p>
--	---

	<p>Teraslinna '492 at 16:42-51: As indicated by block 150, the method includes a step of obtaining a parameter indicative of bandwidth usage associated with the endpoint over at least two of the virtual connections. The parameter provides a measure of bandwidth usage associated with a bundle of provisioned virtual connections which include the at least two of the virtual connections. As previously mentioned, the parameter may be a measure of average bit rate or peak bit rate, and may be computed using a leaky bucket algorithm.</p> <p>Teraslinna '492 at 16:51-58: If the parameter violates a predetermined threshold specified by a predetermined traffic contract, as determined by block 152, a step of inhibiting the flow of packets being forwarded to the endpoint is performed, as indicated by block 154. Consequently, when the flow is inhibited, the number of packets stored in the queue increases as further packets are provided by the customer premises equipment.</p> <p>Teraslinna '492 at 16:64-17:8: If the parameter does not violate the predetermined threshold, i.e., if the parameter complies with the predetermined threshold as determined by block 152, a step of forwarding one of the at least one packet in the queue is performed, as indicated by block 160. A step of updating the parameter when the packet is forwarded to the endpoint from the customer premise equipment is performed, as indicated by block 162. If a leaky bucket parameter is employed, the parameter is incremented in response to the packet being forwarded to the endpoint. Further, the parameter may be decremented in accordance with a step of decrementing the parameter at periodic intervals.</p>
--	--

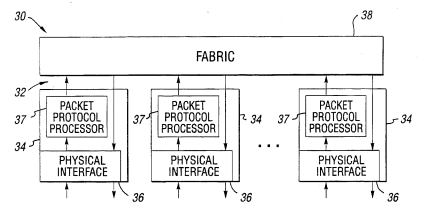
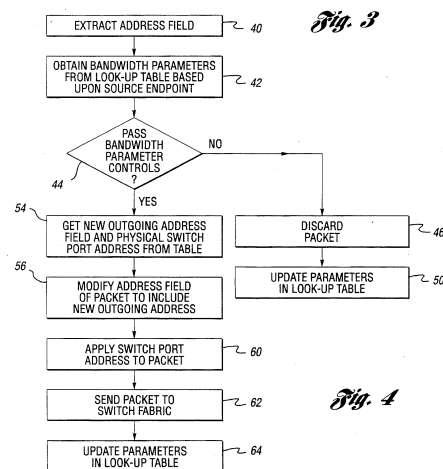
Teraslinna '492 at Fig. 11-12:



Teraslinna '492 at Fig. 2a-2b:



Teraslinna '492 at Fig. 3-4:



	<p>Lerner '776 at [0063]:</p> <p>First, at step 501 the user terminal client web browser 412 submits an HTTP request with a cookie to a web application 401 such that the URL of the desired resource (e.g., application 401) available in the central server domain and the cookie is transmitted to the application 401 . Having received the HTTP request and the cookie, at step 502 the application corresponding to the HTTP request (e.g., application 401) transmits the cookie to the application interface library 402 . The application interface library 402 at step 503 then performs user authentication processing. In particular, the application interface library 402 transmits the cookie received from the client web browser 412 to the cookie access library 405 which is configured to read the cookie data at step 504 . In the event that the transmitted cookie is encrypted, the cookie access library 405 transmits the encrypted cookie to the encryption/ decryption engine 403 which, at step 505 decrypts the encrypted cookie and passes the decrypted cookie back to the cookie access library 405 . The cookie access library 405 then parses the decrypted cookie at step 506 into cookie data and accordingly, determines whether the user ID is matched or not by returning a confirm authentication or a null match command to the application interface library 402 . With a matched user ID, the application interface library 402 returns the result to the web application 401 which then delivers the application content at step 507 for display at the client browser 412 at step 508 . On the other hand, if the authentication process is unsuccessful, a null match command is returned to the web application 401 which then delivers redirect information with a HTTP redirect to the client web browser 412 at step 509 . The client web browser 412 then submits the HTTP redirect request to the login page of the application at step 510 , and the login page URL and the cookie is sent to the login page at the client web browser 412 at step 511 where the user is prompted to input login information to access the desired application.</p> <p>Lerner '776 at [0064]:</p> <p>FIG. 6 illustrates a flowchart for a user event processing procedure to access a desired application in accordance with one embodiment of the present invention. As shown, when a user event occurs such as when a user/client requests access to a particular application or resource in the central server domain, at step 601 the user at the user terminal web browser 412 submits an HTTP request corresponding to the requested application or resource, transmitting the corresponding URL and the cookie file. At step 602 , the corresponding web application 401 receives the HTTP request and the cookie file, and in turn, passes the event name and the</p>
--	---

	<p>received cookie to the application interface library 402 which posts the corresponding event at step 603 . In other words, at step 603 , the requested application or data store from the application interface library 402 places the message signifying the user event onto either a browser cookie or a message queue for eventual delivery at step 607 to another (or subsequent) data store or application. Again, the application, event and the user ID in the form of a message can be optionally encrypted at step 608 by the encryption engine 409 when they are placed onto the message queue before being transmitted from the message queuing middleware 404 of the application or data store resident locally at the central server or at a third party vendor site within the central server domain.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[1.3] at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules; and</p>	<p>The Appendix References disclose and/or render obvious: at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules; and.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 3: Accordingly, a security architecture has been developed in which a single sign-on is provided. Session credentials are used to maintain continuity of a persistent session across multiple accesses to one or more information resources, and in some embodiments, across credential level changes. Session credentials are secured, e.g., as a cryptographically secured session token, such that they may be inspected by a wide variety of entities or applications to verify an authenticated trust level, yet may not be prepared or altered except by a trusted authentication service. Some embodiments of the present invention associate trust level requirements with information resources. Authentication schemes (e.g., those based on passwords, certificates, biometric techniques, smart cards, etc.) are associated with trust levels, and in some embodiments, with environmental parameters. For example, in one configuration, a login service obtains login credentials for an entity commensurate with the trust level requirement(s) of an information resource (or information resources) to be accessed and with environment parameters that affect the sufficiency of a given credential type. Once login credentials have been obtained for an entity and have been authenticated to a given trust level, session credentials are issued and access is</p>

	<p>granted to information resources for which the trust level is sufficient. Advantageously, by using the session credentials access is granted without the need for further login credentials and authentication. In some configurations, session credentials evidencing an insufficient trust level may be remedied by a session continuity preserving upgrade of login credential.</p> <p>Wood at p. 5: Entity: A user or object, including data objects and/or functional objects such as applications, components, modules, services, processes, threads, etc., as well as instantiations thereof. In some configurations, only user entities (typically, the human principal interacting with a software program or on whose behalf a software agent purports to act) are considered. In other configurations, entities include functional objects without an associated human principal. The identity of an entity may be authenticated using a credential.</p> <p>Wood at p. 6: FIG. 1 provides an overview of major interactions between components for an exemplary security architecture in accordance with the present invention. As illustrated in FIG. 1, a client application, e.g., a browser 170 operated by a user, interacts with the security architecture via a gatekeeper and entry handler component 110 and a login component 120. Gatekeeper and entry handler component 110 provides an entry point for external client applications requesting access to enterprise applications and/or resources, including e.g., information resources 191, 192 193, for which access management is provided by the security architecture. Using facilities provided by a session management component 160, an authorization component 140, an authentication component 130, an identification component 150, and login component 120, the gatekeeper/entry handler component 110 allows, redirects or refuses access requests in accordance with a security policy.</p> <p>Wood at p. 6: Individual information resources typically have differing security requirements. In addition, individual types of access to a single information resource may have differing security requirements. Nonetheless, a given level of security may be sufficient for more than one of the information services or access types. For example, information resource 191 may include a product information service for providing general information such as product descriptions or data sheets to the public, while information resource 192 includes an order processing system for an eCommerce site. Information resource 193 may include functions for supply chain</p>
--	---

	<p>interactions such as access to inventory information or current selling price information. Both the product information service and order intake functions of the eCommerce may operate with similar security requirements, e.g., allowing access by minimally authenticated, non-hostile entities. On the other hand, supply chain functions may require a higher level of security. Order status functions of the order processing system may require a mid-level of security.</p> <p>Wood at p. 7: Of note, in certain embodiments in accordance with the present invention, the mapping of login credential types and authentication mechanisms to trust levels is influenced by environment information such as time of request, source of request, connection speed, and/or client application (e.g., browser) environment information. In this way, even with a static set of mapping rules, the set of credential types and authentication mechanisms suitable to support a given trust level may vary based on environment information. In general, mapping rule dependencies are based on perceived variations in threat characteristics and/or requesting entity capabilities. In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 is the authority on environment information for a particular requesting entity.</p> <p>Wood at p. 7: In some embodiments, mapping rules may be dynamically varied. For example, if a particular login credential type and/or authentication method is deemed insecure (e.g., because compromised or because of a changing threat profile), the trust level mappings can be updated and have enterprise-wide effect. In addition, several other advantages are achieved by defining the authentication requirement interface between enterprise applications and/or resources and the security architecture in terms of required trust levels, rather than in terms of particular credential types and authentication methods. First, single sign-on configurations are facilitated using an enterprise-wide credential obtaining, authentication and session tracking infrastructure. Second, authentication requirements may be enforced uniformly in accordance with an enterprise-wide security policy and with reduced vulnerability to a lax security implementation by any particular information resource. Third, credential types and authentication methods can be added, deleted, or mapped to a new trust level, all without modification of enterprise applications and resources. Of course, all advantages need not be achieved in any particular implementation.</p>
--	---

	<p>Wood at p. 8:</p> <p>In addition to the obtained login credentials, some configurations in accordance with the present invention employ session credentials as a mechanism for evidencing prior authentication of obtained login credentials and for binding individual transactions to a particular session. In some configurations, session credentials are also employed in a session token form advantageous for transactions external to the security architecture. In an exemplary realization, session tokens are encoded for supply to browsers as cookies. FIG. 4 illustrates relationships between exemplary login credential, session credential and session token objects.</p> <p>Wood at p. 9:</p> <p>Referring again to session credentials structure 420, a session id, a principal id, a trust level, group ids, a creation time and an expiration time are encoded in both in encrypted portion 421 and clear text portion 422. The session id is a unique identifier for a persistent session maintained by the security architecture. In implementations in which credential upgrade is provided or in which a session credential expiration and refresh is provided, multiple successively issued session credential instances may encode the same session id and correspond to the same persistent session. Principal id encodes an identifier for a principal to which the security architecture has resolved login credentials. For example, a login credential including a username jdoe and a password corresponding to jdoe may be resolved by the security architecture to a unique principal id associated with John. Q. Doe of the shipping and receiving department, having an employee badge number of 12345, etc.</p> <p>Wood at p. 9:</p> <p>In some embodiments, a trust level encodes the authorization level to which a principal has been authenticated. In such embodiments, the encoded trust level serves as a basis for evaluating whether a principal associated with the session credentials has been authenticated to a sufficient level for access to a requested resource. For example, a trust level of 5 may be sufficient for access to information resources having a trust level requirement of 5 or less. Trust levels offer an attractive decoupling of authorization levels and authentication methods as described elsewhere herein. However, in some embodiments, an authorization encoding may establish that a principal has been authenticated using a particular authentication mechanism. In either case, an authorization (e.g., encoded as a trust level) in a cryptographically secured session credential</p>
--	---

	<p>allows the security architecture to authorize accesses based on prior authentication of a login credential and without involvement of the authentication service.</p> <p>Wood at p. 10-11: Focusing then on an exemplary browser-type client entity, browser 170 requests access to one or more of enterprise applications and/or resources (e.g., information resource 191) by presenting an URL to gatekeeper/entry handler component 110, which acts as a point of entry for client entities requesting access to applications and/or resources controlled by the security architecture. Gatekeeper/entry handler component 110 receives the request as a proxy for the requested resource. In some configurations, a combined gatekeeper/entry handler instance is provided, while in others, separate and/or multiple instances are provided with functionally identical interfaces to other components of the security architecture. In some configurations, multiple instances of entry handler functionality (e.g., interception of inbound requests and collection of environment information) are provided. For example, one or more instances for each of several connection types (e.g., dialup, WAN, etc.) may be employed. One or more instances of gatekeeper functionality (e.g., allowing access for authorized requests and otherwise denying or initiating appropriate responsive action) may also be provided. Configurations and functional decompositions suitable to a particular environment and expected load requirements will be appreciated by persons of ordinary skill in the art.</p> <p>Wood at p. 11: Gatekeeper functionality (e.g., in gatekeeper/entry handler component 110) checks whether a session is already associated with the incoming request. Although other techniques are possible, in some configurations in accordance with the present invention, gatekeeper/entry handler component 110 checks for the presence of a session token in the incoming request. Use of session tokens is described in greater detail below; however, in short, a session token may be any data supplied to the client entity for use in uniquely identifying an associated session. In general, preferred session token implementations are cryptographically secured and include facilities, such as expiration or mapping to a particular connection, to limit risk of replay attack and/or spoofing. Some session token implementations may encode session, principal, and/or trust level information. Some session token implementations may employ cookies, URL encoding, or other similar techniques for binding to incoming requests.</p>
--	---

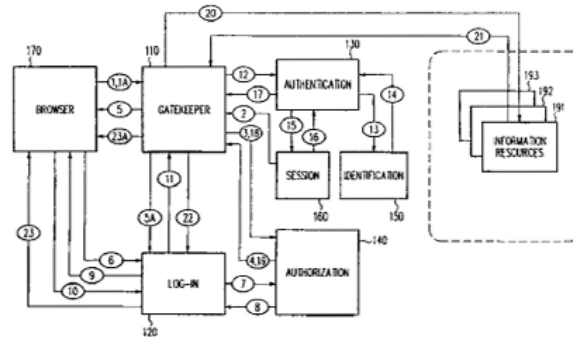
	<p>Wood at p. 14: Generally, mapping rule logic is typically evaluated before a user is challenged to authenticate. Mapping occurs as a function of session environment and particulars of the information resource for which access is requested. By evaluating the minimum trust level required by the target of an access request, a service (e.g., a login service such as provided by login component 120) derives a list of potential authentication methods. The service then checks current session environment against the allowed environment states for each potential authentication method to trim the list further. If there is no particular resource for which access is being requested (e.g., if a user jumps straight to a sign-on page without requesting an access), the service will proceed according to the lowest level of trust available consistent with session environment. Other configurations may employ differing default behaviors.</p> <p>Wood at p. 15-16: Login component 120 also passes control of the access request back to gatekeeper/entry handler component 110. In an exemplary configuration, login component 120 issues a new HTTP request (11) specifying the originally requested URL, which has been stored in the HttpSession object. As before, gatekeeper/entry handler component 110 receives the request. Gatekeeper/entry handler component 110 extracts the login credentials from the request or from the HttpSession object and passes (12) the login credentials to authentication component 130 for authentication. Authentication component 130 authenticates the login credential, and if successful, queries (13) identification component 150 to establish a correspondence with a set of entity descriptors that uniquely identifies the requesting entity. In an exemplary configuration, entity descriptor types include: entity id, system id (e.g., name/password), certificate, enigma id, smartcard token, name/address, and phone. One or more of values may uniquely identify an entity and one or more entity descriptor types may support multiple values (e.g., multiple digital certificates, name/password pairs, or phone numbers per identity). Once the requesting entity has been identified (14), session parameters are updated (15) and session management component 160 supplies (16) session credentials. Preferably, session credentials are digitally-signed or otherwise cryptographically secured and returned (17) to gatekeeper/entry handler component 110.</p> <p>Wood at p. 16: Gatekeeper/entry handler component 110 again supplies (18) authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the</p>
--	--

	<p>associated session (e.g., the signed session credentials. authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency the session credentials (and underlying authentication of login credentials) for the trust level required for the requested access. Login credentials should now be sufficient for access to the requested resource and an ALLOW response is supplied (19) by authorization component 140. Gatekeeper/entry handler component 110 proxies the requested access (20, 21) to information resource 191 and streams (22) results back to login component 120. Login component 120, in turn, streams (23) results back to browser 170.</p> <p>Wood at p. 16-17:</p> <p>Referring again to FIG. 1, subsequent access requests (e.g., access request 1A) include a previously assigned session token. As described above, gatekeeper/entry handler component 110 uses the session token, if provided to resolve a session object containing session credentials, and to determine whether previously authenticated credentials are sufficient for the requested access. As described above, authorization component 140 may be queried using session credentials and an identifier for the requested resource to determine sufficiency of previously authenticated credentials. In some configurations, sufficiency is determined using trust level mappings as described above. Depending on the information resource to which access is requested, and in some configurations depending on current session environment information, access request 1A may or may not have associated previously authenticated credentials sufficient to support the requested access. In the case of an access request 1A having a trust level requirement commensurate with previously obtained and authenticated credentials (i.e., an access request for which no additional credentials need be obtained via login component 120), the access request is proxied (20) and results (21) are streamed directly (23A) back to browser 170. In some configurations, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated session token, if supplied, resolves to the same session object as the session token replaced. For example, in some configurations, both session tokens encode a same session handle, although other aspects associated with session token security such as expiration may be updated. In other configurations, results (21) are streamed (23A) back to browser 170 without an updated session token. In such configurations, the previously supplied session token remains valid until expired or otherwise invalidated. Some variations may employ a session token refresh interval.</p>
--	--

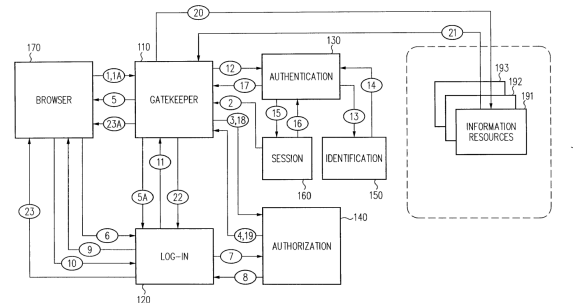
	<p>Wood at p. 18:</p> <p>FIG. 2 illustrates operation of an exemplary security architecture providing a single sign-on facility with trust level mapping to authentication requirements. As before, an access request is received (201) from a client entity. If the request does not contain a session identifier (e.g., a session key or token) or if the request can otherwise be reliably associated with a session maintained by the security architecture, a new session is created (202). A trust level requirement is determined for access to the requested resource in the context of the requesting session environment. In some configurations, as described above, the determination is performed by an authorization service such as authorization component 140. Given a trust level requirement, current session credentials are evaluated (203) in the context of session environment information to determine whether the previously supplied login credentials are sufficient to achieve the required trust level. In one advantageous realization of session credentials and tokens, a cryptographically secured encoding of trust level allows authorization to be confirmed without involvement of an authentication service (e.g., with reauthentication of login credentials). In the case of a newly created (202) session, the authorization evidenced by session credentials will typically be insufficient, although some security policies may allow anonymous access to certain resources.</p> <p>Wood at p. 19:</p> <p>Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information. Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents</p>
--	---

back to the credential gathering service. Others (e.g., certificates) may be obtained for the client entity from an agent or authority. For example, a personal digital certificate (such as those issued by VerisignTM, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select 25 from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.

Wood at Fig. 1:



Wood at Fig. 2:



	<p>Teraslinna '492 at Abstract: Methods and systems for resource management in a fast packet switching communication network are disclosed. A packet containing an address field identifying a source endpoint and a destination endpoint is received. A bandwidth constraint based upon the source endpoint, but independent of the destination endpoint, is enforced for the packet. In a communication network wherein a plurality of virtual connections are provisioned from a single source endpoint to a plurality of destination endpoints, the bandwidth constraint is associated with bandwidth usage over all of the virtual connections.</p> <p>Teraslinna '492 at 1:9-13: The present invention relates generally to managing bandwidth resources within a switching communication network, and more particularly, to managing bandwidth resources within a switching communication network having at least one fast packet switch.</p> <p>Teraslinna '492 at 1:17-34: Fast packet switching is an emerging multiplexing and switching technique used in cell relay services, such as the asynchronous transfer mode (ATM) technique, and frame relay services. Briefly, fast packet switching is a technique wherein a message is communicated between a source endpoint and a destination endpoint of a switching network using a series of information passing elements, i.e. packets, each of which contain both a payload field and an address field. The switching network used for connecting the two endpoints is composed of one or more fast packet switches with transmission facilities connecting the switches. Each of the fast packet switches contains a plurality of switch ports between which packets are communicated via a self-routing switching fabric. As a result, each of the switch ports can be employed either as a connection to an endpoint or as a connection to another fast packet switch in the network via the transmission facilities.</p> <p>Teraslinna '492 at 1:35-47: Each message to be communicated using fast packet switching is packetized by dividing the message into one or more pieces of information. Each of the pieces of information is loaded into the payload field of a corresponding packet. In a frame relay service, the packets are referred to as frames, and the length of the payload field is variable. In contrast, the length of the payload field is fixed in a cell relay service, wherein the packets are referred to as cells. Regardless of the type of packet employed, each of the packets is further loaded with an</p>
--	---

	<p>address field which contains information identifying a provisioned virtual connection between the two endpoints. Typically, the address field is contained in a header in the packet.</p> <p>Teraslinna '492 at 1:48-57: The packets are then forwarded to the switching network at the source endpoint for transmission to the destination endpoint. The switching network routes the packets to the destination endpoint based upon routing data in the fast packet switches. The routing data is written into the fast packet switches by a connection management processor at the time the virtual connection is provisioned. In particular, the routing data specifies how a packet is to be routed through the network based upon information contained within the address field of the packet.</p> <p>Teraslinna '492 at 4:1-10: FIG. 1 is a flow chart of an embodiment of a method of resource management in a fast packet switching network. The fast packet switching network, having a plurality of endpoints, includes at least one fast packet switch used in providing a virtual connection between at least one selected pair of the endpoints. The method includes a step of receiving a packet having an address field which identifies a source endpoint and a destination endpoint, as indicated by block 10. Typically, the step of receiving the packet is performed at a switch port in a fast packet switch within the network.</p> <p>Teraslinna '492 at 4:39-46: An example of a packet 20 having a protocol in accordance with the present invention is illustrated in FIG. 2a. The packet 20 includes a payload field 21 and an address field 22. The address field 22 includes a source-associated part 23 and a destination-associated part 24. The address field 22 is contained within a header of the packet 20. Hereinafter, an address field in accordance with this protocol is referred to as a virtual connection identifier label (VCIL).</p> <p>Teraslinna '492 at 5:1-10: Referring back to FIG. 1, the method further includes a step of enforcing a bandwidth constraint, as indicated by block 12, based upon the source endpoint identified by the address field, wherein the bandwidth constraint is independent of the destination endpoint identified by the address field. In a preferred embodiment wherein the address field contains a source label which identifies the source endpoint and a destination label which identifies the destination</p>
--	--

	<p>endpoint, the bandwidth constraint is enforced based upon the source label and is independent of the destination label.</p> <p>Teraslinna '492 at 5:26-38: Based upon whether or not the packet violates the bandwidth constraint and the type of penalty employed for violating packets, the method may further include a step of routing the packet based upon the address field, as indicated by block 14. Here, the packet is routed from the switch port at which the packet is received, through the switching fabric contained within the fast packet switch, to a second port for transmission toward the destination endpoint. As one with ordinary skill in the art will recognize, the destination endpoint may be located at a switch port on another fast packet switch connected to the second port via a transmission facility, and as such may not be directly interfaced to the second port.</p> <p>Teraslinna '492 at 6:45-49: FIG. 4 is a flow chart of a method of processing a received packet for use with embodiments of the line unit in accordance with the present invention. Upon receiving the packet, the address field, typically located in a header in the packet, is extracted as indicated by block 40.</p> <p>Teraslinna '492 at 15:40-49: The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint 132 over at least two of the virtual connections. The at least one parameter indicative of bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.</p> <p>Teraslinna '492 at 15:50-59: The at least one parameter parallels at least one of the bandwidth parameters monitored and enforced by the network 134 in accordance with a predetermined traffic contract. Preferably, the traffic flow processor 142 obtains each of the parameters which are monitored by the network 134 in accordance with the predetermined traffic contract. The at least one parameter may be indicative of an average bit rate or a peak bit rate, for example, associated with the bundle. The at least one parameter may include a leaky bucket parameter as described earlier.</p>
--	--

	<p>Teraslinna '492 at 16:64-17:8:</p> <p>If the parameter does not violate the predetermined threshold, i.e., if the parameter complies with the predetermined threshold as determined by block 152, a step of forwarding one of the at least one packet in the queue is performed, as indicated by block 160. A step of updating the parameter when the packet is forwarded to the endpoint from the customer premise equipment is performed, as indicated by block 162. If a leaky bucket parameter is employed, the parameter is incremented in response to the packet being forwarded to the endpoint. Further, the parameter may be decremented in accordance with a step of decrementing the parameter at periodic intervals.</p>
--	---

Teraslinna '492 at Fig. 11-12:

U.S. Patent Apr. 22, 1997 Sheet 6 of 6 5,623,492

Fig. 11

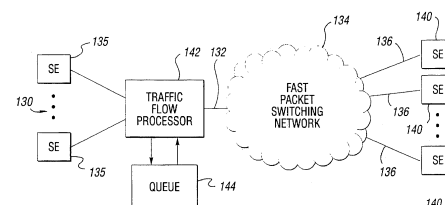
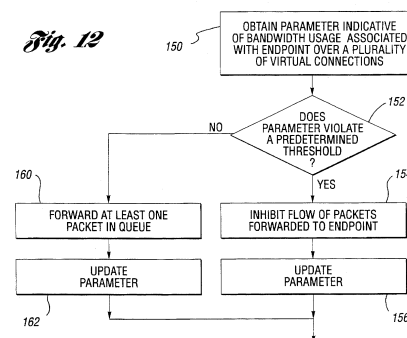
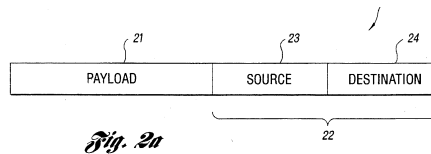
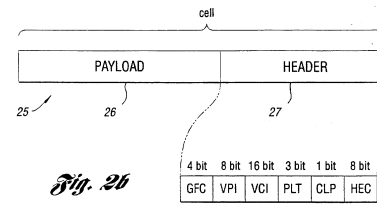


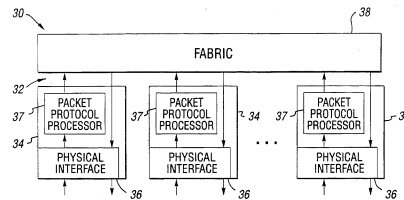
Fig. 12



Teraslinna '492 at Fig. 2a-2b:



Teraslinna '492 at Fig. 3:



Lerner '776 at [0025]:

FIG. 7 illustrates a flowchart for a user event processing of a central server in accordance with one embodiment of the present invention.

Lerner '776 at [0064]:

FIG. 6 illustrates a flowchart for a user event processing procedure to access a desired application in accordance with one embodiment of the present invention. As shown, when a user event occurs such as when a user/client requests access to a particular application or resource in the central server domain, at step 601 the user at the user terminal web browser 412 submits an HTTP request corresponding to the requested application or resource, transmitting

the corresponding URL and the cookie file. At step 602 , the corresponding web application 401 receives the HTTP request and the cookie file, and in turn, passes the event name and the received cookie to the application interface library 402 which posts the corresponding event at step 603 . In other words, at step 603 , the requested application or data store from the application interface library 402 places the message signifying the user event onto either a browser cookie or a message queue for eventual delivery at step 607 to another (or subsequent) data store or application. Again, the application, event and the user ID in the form of a message can be optionally encrypted at step 608 by the encryption engine 409 when they are placed onto the message queue before being transmitted from the message queuing middleware 404 of the application or data store resident locally at the central server or at a third party vendor site within the central server domain.

Lerner '776 at [0065]:

As discussed previously, the application interface library 402 optionally passes the cookie at step 604 to the encryption/ decryption engine 403 as necessary to decrypt the cookie in the event that the received cookie is encrypted. Furthermore, the application interface library 402 passes the user ID, the corresponding event and the cookie to the cookie access library 405 which is configured to update the received cookie at step 605 , and as necessary, encrypt the cookie by passing the updated cookie to the encryption/ decryption engine 403 at step 606 . The updated cookie (optionally encrypted) and the event content is then passed to the user specified web application 401 at step 609 and thereafter, at step 610 , the content corresponding to the user specified web application 401 is displayed on the client browser 412 at the client terminal.

Lerner '776 at [0066]:

FIG. 7 illustrates a flowchart for the user event processing procedure at the central server in accordance with one embodiment of the present invention. As shown, at step 701 the message queuing middleware 408 at the central server receives the message which contains information related to the user specified application, user event and user ID from the message queuing middleware 404 and passes the same to the message broker 410 for parsing the message at step 702 . Optionally, in the case where the received message is previously encrypted (for example, by the encryption engine 403 at step 608 shown in FIG. 6), the encryption engine 409 at the central server terminal decrypts the message at step 703 and returns the decrypted message to

	<p>the message broker 410 . The message broker 410 routes the parsed, decrypted message at step 704 , that is, the message broker 410 determines which applications or data stores, whether local or remote, need to receive the incoming message and routes them accordingly. Thereafter at step 705 , the message broker 410 transforms the message for the respective application or data store into the appropriate format suitable and compatible for the intended application or data store determined at step 704.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[1.4] at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.</p>	<p>The Appendix References disclose and/or render obvious: at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 3: Accordingly, a security architecture has been developed in which a single sign-on is provided. Session credentials are used to maintain continuity of a persistent session across multiple accesses to one or more information resources, and in some embodiments, across credential level changes. Session credentials are secured, e.g., as a cryptographically secured session token, such that they may be inspected by a wide variety of entities or applications to verify an authenticated trust level, yet may not be prepared or altered except by a trusted authentication service. Some embodiments of the present invention associate trust level requirements with information resources. Authentication schemes (e.g., those based on passwords, certificates, biometric techniques, smart cards, etc.) are associated with trust levels, and in some embodiments, with environmental parameters. For example, in one configuration, a login service obtains login credentials for an entity commensurate with the trust level requirement(s) of an information resource (or information resources) to be accessed and with environment parameters that affect the sufficiency of a given credential type. Once login credentials have been obtained for an entity and have been authenticated to a given trust level, session credentials are issued and access is granted to information resources for which the trust level is sufficient. Advantageously, by using</p>

	<p>the session credentials access is granted without the need for further login credentials and authentication. In some configurations, session credentials evidencing an insufficient trust level may be remedied by a session continuity preserving upgrade of login credential.</p> <p>Wood at p. 4: In still yet another embodiment in accordance with the present invention, an information access control facility includes an application proxy and means responsive to the application proxy. The application proxy is for receiving an access request targeting one of the information resources, extracting a cryptographically secured session token from the access request, and selectively proxying the access request. The means responsive to the application proxy is for evaluating sufficiency of an authorization encoded in the cryptographically secured session token for access to the targeted information resource.</p> <p>Wood at p. 5: Access Management: Systems, methods and techniques for controlling use of information resources. Typically, access management systems employ both authentication and authorization to control access to information resources.</p> <p>Wood at p. 5: Entity: A user or object, including data objects and/or functional objects such as applications, components, modules, services, processes, threads, etc., as well as instantiations thereof. In some configurations, only user entities (typically, the human principal interacting with a software program or on whose behalf a software agent purports to act) are considered. In other configurations, entities include functional objects without an associated human principal. The identity of an entity may be authenticated using a credential.</p> <p>Wood at p. 6: FIG. 1 provides an overview of major interactions between components for an exemplary security architecture in accordance with the present invention. As illustrated in FIG. 1, a client application, e.g., a browser 170 operated by a user, interacts with the security architecture via a gatekeeper and entry handler component 110 and a login component 120. Gatekeeper and entry handler component 110 provides an entry point for external client applications requesting access to enterprise applications and/or resources, including e.g., information resources 191, 192 193, for which access management is provided by the security architecture. Using facilities provided</p>
--	---

	<p>by a session management component 160, an authorization component 140, an authentication component 130, an identification component 150, and login component 120, the gatekeeper/entry handler component 110 allows, redirects or refuses access requests in accordance with a security policy.</p> <p>Wood at p. 6: Individual information resources typically have differing security requirements. In addition, individual types of access to a single information resource may have differing security requirements. Nonetheless, a given level of security may be sufficient for more than one of the information services or access types. For example, information resource 191 may include a product information service for providing general information such as product descriptions or data sheets to the public, while information resource 192 includes an order processing system for an eCommerce site. Information resource 193 may include functions for supply chain interactions such as access to inventory information or current selling price information. Both the product information service and order intake functions of the eCommerce may operate with similar security requirements, e.g., allowing access by minimally authenticated, non-hostile entities. On the other hand, supply chain functions may require a higher level of security. Order status functions of the order processing system may require a mid-level of security.</p> <p>Wood at p. 7: Of note, in certain embodiments in accordance with the present invention, the mapping of login credential types and authentication mechanisms to trust levels is influenced by environment information such as time of request, source of request, connection speed, and/or client application (e.g., browser) environment information. In this way, even with a static set of mapping rules, the set of credential types and authentication mechanisms suitable to support a given trust level may vary based on environment information. In general, mapping rule dependencies are based on perceived variations in threat characteristics and/or requesting entity capabilities. In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 is the authority on environment information for a particular requesting entity.</p> <p>Wood at p. 7: In some embodiments, mapping rules may be dynamically varied. For example, if a particular login credential type and/or authentication method is deemed insecure (e.g., because</p>
--	--

	<p>compromised or because of a changing threat profile), the trust level mappings can be updated and have enterprise-wide effect. In addition, several other advantages are achieved by defining the authentication requirement interface between enterprise applications and/or resources and the security architecture in terms of required trust levels, rather than in terms of particular credential types and authentication methods. First, single sign-on configurations are facilitated using an enterprise-wide credential obtaining, authentication and session tracking infrastructure. Second, authentication requirements may be enforced uniformly in accordance with an enterprise-wide security policy and with reduced vulnerability to a lax security implementation by any particular information resource. Third, credential types and authentication methods can be added, deleted, or mapped to a new trust level, all without modification of enterprise applications and resources. Of course, all advantages need not be achieved in any particular implementation.</p> <p>Wood at p. 9: Referring again to session credentials structure 420, a session id, a principal id, a trust level, group ids, a creation time and an expiration time are encoded in both in encrypted portion 421 and clear text portion 422. The session id is a unique identifier for a persistent session maintained by the security architecture. In implementations in which credential upgrade is provided or in which a session credential expiration and refresh is provided, multiple successively issued session credential instances may encode the same session id and correspond to the same persistent session. Principal id encodes an identifier for a principal to which the security architecture has resolved login credentials. For example, a login credential including a username jdoe and a password corresponding to jdoe may be resolved by the security architecture to a unique principal id associated with John. Q. Doe of the shipping and receiving department, having an employee badge number of 12345, etc.</p> <p>Wood at p. 9: In some embodiments, a trust level encodes the authorization level to which a principal has been authenticated. In such embodiments, the encoded trust level serves as a basis for evaluating whether a principal associated with the session credentials has been authenticated to a sufficient level for access to a requested resource. For example, a trust level of 5 may be sufficient for access to information resources having a trust level requirement of 5 or less. Trust levels offer an attractive decoupling of authorization levels and authentication methods as described</p>
--	--

	<p>elsewhere herein. However, in some embodiments, an authorization encoding may establish that a principal has been authenticated using a particular authentication mechanism. In either case, an authorization (e.g., encoded as a trust level) in a cryptographically secured session credential allows the security architecture to authorize accesses based on prior authentication of a login credential and without involvement of the authentication service.</p> <p>Wood at p. 10-11: Focusing then on an exemplary browser-type client entity, browser 170 requests access to one or more of enterprise applications and/or resources (e.g., information resource 191) by presenting an URL to gatekeeper/entry handler component 110, which acts as a point of entry for client entities requesting access to applications and/or resources controlled by the security architecture. Gatekeeper/entry handler component 110 receives the request as a proxy for the requested resource. In some configurations, a combined gatekeeper/entry handler instance is provided, while in others, separate and/or multiple instances are provided with functionally identical interfaces to other components of the security architecture. In some configurations, multiple instances of entry handler functionality (e.g., interception of inbound requests and collection of environment information) are provided. For example, one or more instances for each of several connection types (e.g., dialup, WAN, etc.) may be employed. One or more instances of gatekeeper functionality (e.g., allowing access for authorized requests and otherwise denying or initiating appropriate responsive action) may also be provided. Configurations and functional decompositions suitable to a particular environment and expected load requirements will be appreciated by persons of ordinary skill in the art.</p> <p>Wood at p. 11: Entry handler functionality (e.g., in gatekeeper/entry handler component 110) ascertains much of the requesting client's environment information. For example, for dial-up connections, environment information such as line speed and low-level line encryption are ascertained. Additional information, such as source number (e.g., from caller id information or based on a callback configuration), signaling type (e.g., POTS or digital ISDN), etc., may also be obtained. For network connections, similar environment information (e.g., source network and/or node, Virtual Private Network (VPN) low-level encryption, etc.) may be obtained from incoming requests themselves or based on a particular entry point (e.g., a particular router or port). More generally, gatekeeper/entry handler component 110 obtains and/or maintains information such</p>
--	---

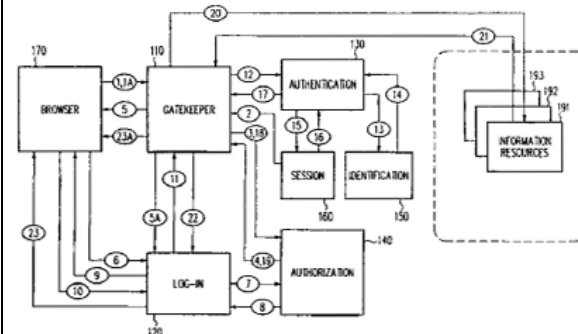
	<p>as connect location (if ascertainable), temporal information such as request time/date, session start time/date, etc. (preferably in both the client entity's frame of reference and the security architecture or requested information resource's frame of reference, if location is ascertainable), and client type and/or capabilities (e.g., browser type and Java Development Kit (JDK) level). In some configurations, information on line speed, origination point (e.g., inside or outside of a corporate network), browser type, encryption capability, number of hops, latency, system type, etc. may be gathered. Such information is used in some configurations for mapping particular authentication mechanisms to trust levels and for authorization decisions. Environment information is generally packaged into a data structure that is associated with a client session. Other components of the security architecture may add additional client environment information, such as authentication strength or current trust level.</p> <p>Wood at p. 14: Generally, mapping rule logic is typically evaluated before a user is challenged to authenticate. Mapping occurs as a function of session environment and particulars of the information resource for which access is requested. By evaluating the minimum trust level required by the target of an access request, a service (e.g., a login service such as provided by login component 120) derives a list of potential authentication methods. The service then checks current session environment against the allowed environment states for each potential authentication method to trim the list further. If there is no particular resource for which access is being requested (e.g., if a user jumps straight to a sign-on page without requesting an access), the service will proceed according to the lowest level of trust available consistent with session environment. Other configurations may employ differing default behaviors.</p> <p>Wood at p. 15-16: Login component 120 also passes control of the access request back to gatekeeper/entry handler component 110. In an exemplary configuration, login component 120 issues a new HTTP request (11) specifying the originally requested URL, which has been stored in the HttpSession object. As before, gatekeeper/entry handler component 110 receives the request. Gatekeeper/entry handler component 110 extracts the login credentials from the request or from the HttpSession object and passes (12) the login credentials to authentication component 130 for authentication. Authentication component 130 authenticates the login credential, and if successful, queries (13) identification component 150 to establish a correspondence with a set of entity descriptors that</p>
--	---

	<p>uniquely identifies the requesting entity. In an exemplary configuration, entity descriptor types include: entity id, system id (e.g., name/password), certificate, enigma id, smartcard token, name/address, and phone. One or more of values may uniquely identify an entity and one or more entity descriptor types may support multiple values (e.g., multiple digital certificates, name/password pairs, or phone numbers per identity). Once the requesting entity has been identified (14), session parameters are updated (15) and session management component 160 supplies (16) session credentials. Preferably, session credentials are digitally-signed or otherwise cryptographically secured and returned (17) to gatekeeper/entry handler component 110.</p> <p>Wood at p. 16: Gatekeeper/entry handler component 110 again supplies (18) authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session (e.g., the signed session credentials. authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency the session credentials (and underlying authentication of login credentials) for the trust level required for the requested access. Login credentials should now be sufficient for access to the requested resource and an ALLOW response is supplied (19) by authorization component 140. Gatekeeper/entry handler component 110 proxies the requested access (20, 21) to information resource 191 and streams (22) results back to login component 120. Login component 120, in turn, streams (23) results back to browser 170.</p> <p>Wood at p. 16-17: Referring again to FIG. 1, subsequent access requests (e.g., access request 1A) include a previously assigned session token. As described above, gatekeeper/entry handler component 110 uses the session token, if provided to resolve a session object containing session credentials, and to determine whether previously authenticated credentials are sufficient for the requested access. As described above, authorization component 140 may be queried using session credentials and an identifier for the requested resource to determine sufficiency of previously authenticated credentials. In some configurations, sufficiency is determined using trust level mappings as described above. Depending on the information resource to which access is requested, and in some configurations depending on current session environment information, access request 1A may or may not have associated previously authenticated credentials sufficient to support the requested access. In the case of an access request 1A having a trust level requirement</p>
--	--

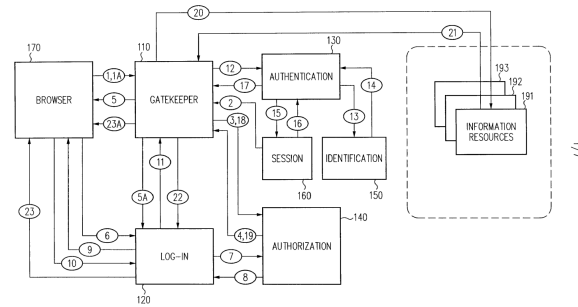
	<p>commensurate with previously obtained and authenticated credentials (i.e., an access request for which no additional credentials need be obtained via login component 120), the access request is proxied (20) and results (21) are streamed directly (23A) back to browser 170. In some configurations, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated session token, if supplied, resolves to the same session object as the session token replaced. For example, in some configurations, both session tokens encode a same session handle, although other aspects associated with session token security such as expiration may be updated. In other configurations, results (21) are streamed (23A) back to browser 170 without an updated session token. In such configurations, the previously supplied session token remains valid until expired or otherwise invalidated. Some variations may employ a session token refresh interval.</p> <p>Wood at p. 18:</p> <p>FIG. 2 illustrates operation of an exemplary security architecture providing a single sign-on facility with trust level mapping to authentication requirements. As before, an access request is received (201) from a client entity. If the request does not contain a session identifier (e.g., a session key or token) or if the request can otherwise be reliably associated with a session maintained by the security architecture, a new session is created (202). A trust level requirement is determined for access to the requested resource in the context of the requesting session environment. In some configurations, as described above, the determination is performed by an authorization service such as authorization component 140. Given a trust level requirement, current session credentials are evaluated (203) in the context of session environment information to determine whether the previously supplied login credentials are sufficient to achieve the required trust level. In one advantageous realization of session credentials and tokens, a cryptographically secured encoding of trust level allows authorization to be confirmed without involvement of an authentication service (e.g., with reauthentication of login credentials). In the case of a newly created (202) session, the authorization evidenced by session credentials will typically be insufficient, although some security policies may allow anonymous access to certain resources.</p> <p>Wood at p. 19:</p> <p>Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The</p>
--	--

credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information. Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents back to the credential gathering service. Others (e.g., certificates) may be obtained for the client entity from an agent or authority. For example, a personal digital certificate (such as those issued by Verisign™, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.

Wood at Fig. 1:



Wood at Fig. 2:



Teraslinna '492 at 7:64-8:9:

Using embodiments of the present invention, bandwidth use is monitored and enforced for each of the three source endpoints 82a-82c rather than for each of the 15 virtual connections. In terms of the bundling terminology: the five virtual connections from the endpoint 82a are considered as a first bundle; the five virtual connections from the endpoint 82b are considered as a second bundle; and the five virtual connections from the endpoint 82c are considered as a third bundle. As a result, less bandwidth needs to be allocated in the transmission facility 76 to provide the 15 virtual connections in three bundles, in comparison to having 15 provisioned virtual connections each having bandwidth associated therewith.

Teraslinna '492 at 15:40-49:

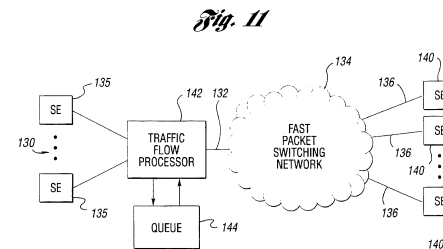
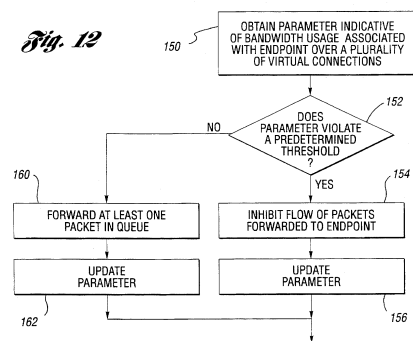
The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint 132 over at least two of the virtual connections. The at least one parameter indicative of bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.

Teraslinna '492 at 15:60-67:

The traffic flow processor 142 further provides means for inhibiting the flow of packets forwarded to the endpoint 132 when the at least one parameter violates the traffic contract. Typically, this occurs when one of the at least one parameter violates a predetermined

	<p>threshold. The predetermined threshold may be violated if the parameter exceeds the threshold, or alternatively, if the parameter drops below the threshold.</p> <p>Teraslinna '492 at 16:1-8: The system further includes a queue 144 in communication with the traffic flow processor 142. The queue 144 provides means for storing at least one packet when the traffic contract is violated, i.e. when the parameter violates the predetermined threshold. Preferably, the queue 144 queues all incoming packets from the customer premises equipment 130 for traffic shaping by the traffic flow processor 142.</p> <p>Teraslinna '492 at 16:13-21: When the at least one parameter complies with the predetermined traffic contract, i.e., each parameter complies with a corresponding predetermined threshold, the traffic flow processor 142 performs a step of forwarding one of the at least one packet stored in the queue 144 to the endpoint 132. The traffic flow processor 142 further updates each of the at least one parameter when the packet is forwarded to the endpoint 132. Typically, the leaky bucket parameter is updated by performing a step of incrementing the parameter.</p>
--	---

Teraslinna '492 at Fig. 11-12:



Teraslinna '492 at Fig. 5:

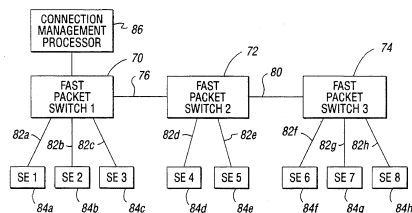


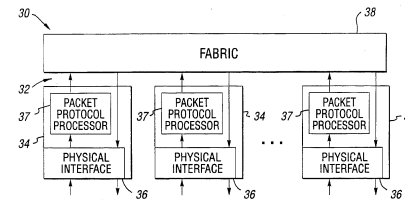
Fig. 5

	<p>Lerner '776 at [0066]:</p> <p>FIG. 7 illustrates a flowchart for the user event processing procedure at the central server in accordance with one embodiment of the present invention. As shown, at step 701 the message queuing middleware 408 at the central server receives the message which contains information related to the user specified application, user event and user ID from the message queuing middleware 404 and passes the same to the message broker 410 for parsing the message at step 702 . Optionally, in the case where the received message is previously encrypted (for example, by the encryption engine 403 at step 608 shown in FIG. 6), the encryption engine 409 at the central server terminal decrypts the message at step 703 and returns the decrypted message to the message broker 410 . The message broker 410 routes the parsed, decrypted message at step 704 , that is, the message broker 410 determines which applications or data stores, whether local or remote, need to receive the incoming message and routes them accordingly. Thereafter at step 705 , the message broker 410 transforms the message for the respective application or data store into the appropriate format suitable and compatible for the intended application or data store determined at step 704.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[2] The method according to claim 1, wherein the plurality of stages further comprises an interface specific stage, further comprising: at the interface specific stage, applying interface specific rules to the packet based on a type of a network interface over which the packet is received.</p>	<p>The Appendix References disclose and/or render obvious: The method according to claim 1, wherein the plurality of stages further comprises an interface specific stage, further comprising: at the interface specific stage, applying interface specific rules to the packet based on a type of a network interface over which the packet is received.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 10-11:</p> <p>Focusing then on an exemplary browser-type client entity, browser 170 requests access to one or more of enterprise applications and/or resources (e.g., information resource 191) by presenting an URL to gatekeeper/entry handler component 110, which acts as a point of entry for client entities requesting access to applications and/or resources controlled by the security architecture. Gatekeeper/entry handler component 110 receives the request as a proxy for the requested resource. In some configurations, a combined gatekeeper/entry handler instance is provided, while in others, separate and/or multiple instances are provided with functionally identical</p>

	<p>interfaces to other components of the security architecture. In some configurations, multiple instances of entry handler functionality (e.g., interception of inbound requests and collection of environment information) are provided. For example, one or more instances for each of several connection types (e.g., dialup, WAN, etc.) may be employed. One or more instances of gatekeeper functionality (e.g., allowing access for authorized requests and otherwise denying or initiating appropriate responsive action) may also be provided. Configurations and functional decompositions suitable to a particular environment and expected load requirements will be appreciated by persons of ordinary skill in the art.</p> <p>Wood at p. 11:</p> <p>Entry handler functionality (e.g., in gatekeeper/entry handler component 110) ascertains much of the requesting client's environment information. For example, for dial-up connections, environment information such as line speed and low-level line encryption are ascertained. Additional information, such as source number (e.g., from caller id information or based on a callback configuration), signaling type (e.g., POTS or digital ISDN), etc., may also be obtained. For network connections, similar environment information (e.g., source network and/or node, Virtual Private Network (VPN) low-level encryption, etc.) may be obtained from incoming requests themselves or based on a particular entry point (e.g., a particular router or port). More generally, gatekeeper/entry handler component 110 obtains and/or maintains information such as connect location (if ascertainable), temporal information such as request time/date, session start time/date, etc. (preferably in both the client entity's frame of reference and the security architecture or requested information resource's frame of reference, if location is ascertainable), and client type and/or capabilities (e.g., browser type and Java Development Kit (JDK) level). In some configurations, information on line speed, origination point (e.g., inside or outside of a corporate network), browser type, encryption capability, number of hops, latency, system type, etc. may be gathered. Such information is used in some configurations for mapping particular authentication mechanisms to trust levels and for authorization decisions. Environment information is generally packaged into a data structure that is associated with a client session. Other components of the security architecture may add additional client environment information, such as authentication strength or current trust level.</p>
--	---

	<p>Wood at p. 16:</p> <p>Although the encoding of session tokens using cookies is presently preferred, in part because cookie facilities are widely supported and reasonably well accepted, other facilities may be employed to establish session continuity. For example, alternative URL encodings and/or custom or plug-in support for session identifier receipt, storage and supply may be employed. Also, some configurations may employ lower-level session identifiers, e.g., as provided by a particular connection type such as trusted caller id information or as associated with a low-level line encryption or virtual private network infrastructure. As such facilities are likely to be connection-type specific, it is envisioned that they may be used in conjunction with other session identifier facilities described above, e.g., session tokens encoded in cookies. In one configuration, the unique Ethernet MAC address associated with a network interface card may be employed as a session handle. The MAC address is then used to associate with a particular set of session credentials maintained by a central authority. In general the representation of a session handle can take many forms.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p> <p>Teraslinna '492 at 6:16-23:</p> <p>Each of the switch ports 32 (or each transmission facility) has a line unit 34 associated therewith. Each line unit 34 performs two main functions: (i) it processes the physical layer protocol, and (ii) it processes the packet protocol. A physical interface 36 performs the physical layer protocol processing. As such, the physical interface 36 provides means for receiving and transmitting electrical signals representative of received and transmitted packets.</p> <p>Teraslinna '492 at 6:24-33:</p> <p>A packet protocol processor 37 performs the packet protocol processing. The packet protocol processor 37 provides means for enforcing a bandwidth constraint based upon an address field contained within the received packet. Consequently, the line unit 34 enforces a bandwidth constraint for packets communicated over a bundle of provisioned virtual connections emanating from a source endpoint identified by the address field. As described earlier, a received packet which violates the bandwidth constraint is penalized in a predetermined manner.</p>
--	--

Teraslinna '492 at Fig. 3:



Lerner '776 at [0026]:

FIG. 1 illustrates an integrated, distributed shared services architecture in accordance with one embodiment of the present invention. As shown, there is provided an application layer 110 , a services layer 130 and an interface layer 120 which resides between the application layer 110 and the services layer 130 . The interface layer 120 is configured to receive requests for services from the various web applications or other central server applications at the application layer 110 , and passes the received requests to the corresponding services at the services layer 130 after performing the necessary protocol, formatting and other transformations. The interface layer 120 is further configured to return any required results back to the calling application at the application layer 110 after performing necessary protocol, formatting or other transformations.

Lerner '776 at [0029]:

To permit proper communication between the applications in the application layer 110 and the services layer 130 , in accordance with one aspect of the present invention, an interface adapter may be used to facilitate application integration as will be discussed in further detail below. In this manner, geographically dispersed and architecturally diverse and potentially changing set of applications can use the integrated, distributed shared services architecture and permit changes to them with the least amount of disruptions and effort.

Lerner '776 at Fig. 1:

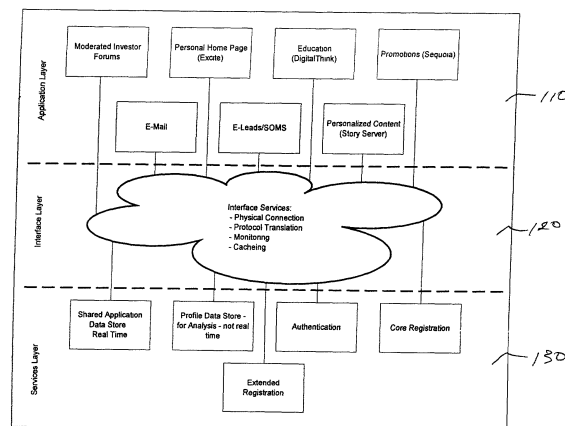


FIGURE 1

[3] The method according to claim 2, wherein the plurality of stages further comprises a global stage, further comprising: at the global stage, applying global firewall rules to the packet.

The Appendix References disclose and/or render obvious: The method according to claim 2, wherein the plurality of stages further comprises a global stage, further comprising: at the global stage, applying global firewall rules to the packet.

See, e.g.:

Wood at p. 10-11:

Focusing then on an exemplary browser-type client entity, browser 170 requests access to one or more of enterprise applications and/or resources (e.g., information resource 191) by presenting an URL to gatekeeper/entry handler component 110, which acts as a point of entry for client entities requesting access to applications and/or resources controlled by the security architecture. Gatekeeper/entry handler component 110 receives the request as a proxy for the requested

	<p>resource. In some configurations, a combined gatekeeper/entry handler instance is provided, while in others, separate and/or multiple instances are provided with functionally identical interfaces to other components of the security architecture. In some configurations, multiple instances of entry handler functionality (e.g., interception of inbound requests and collection of environment information) are provided. For example, one or more instances for each of several connection types (e.g., dialup, WAN, etc.) may be employed. One or more instances of gatekeeper functionality (e.g., allowing access for authorized requests and otherwise denying or initiating appropriate responsive action) may also be provided. Configurations and functional decompositions suitable to a particular environment and expected load requirements will be appreciated by persons of ordinary skill in the art.</p> <p>Wood at p. 12-13:</p> <p>Gatekeeper/entry handler component 110 supplies authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session. Preferably, the associated session identifier is cryptographically secured (e.g., as a signed session credential). In some configurations, the signed session credential is obtained from the corresponding session object. In the case of a pre-existing session, the signed session credential may be obtained using a received session token. In any case, authorization component 140 receives (3) the requested resource and session identifiers and responds (4) in accordance with the trust level requirement of the requested resource. In configurations for which sensitivity to a changing environment is desired, environment information may also be supplied to authorization component 140. In an exemplary configuration, authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency of a current trust level. In some configurations, authorization component 140 dynamically calculates a current trust level based on the signed session credentials and environment information. In others, authorization component 140 may base its ALLOW, REDIRECT, or REFUSE response on a "current" trust level previously associated with the signed session credentials. Generally, an access request with sufficient current trust level is ALLOWED and forwarded (14) without further authentication. An authorization request without proper parameters (e.g., without a specified information resource or without a properly secured session credential) may be REFUSED. Authorization requests associated with a client entity that has been blacklisted or for a resource for which the associated client entity cannot be authenticated using any available method to achieve the required trust level may also be REFUSED. For example, a given security</p>
--	---

	<p>policy and associated trust level mappings may dictate a REFUSE response in response to an access request to a sensitive resource (such as financial data) by any client entity (even a browser supplying the digital certificate for the CFO, and therefore presumably operating on behalf of the CEO) if the access request is received over a dial-up line and originates from an unknown number or is received outside of working hours.</p> <p>Wood at p. 14:</p> <p>Browser 170 sends (6) login component 120 a new access request using the URL specified in the redirect from gatekeeper/entry handler component 110. In configurations employing cookies as a medium for passing session tokens, the new access request will include the cookie and therefore the session token. Note that in configurations in which the security architecture controls access to resources in several domains, care should be exercised to select a tag or tags for the cookie such that it will be provided through normal operation of the browser in subsequent accesses to any of the several domains. Persons of ordinary skill in the art will appreciate suitable tagging techniques, including the use of multiple cookies. Login component 120 receives the access request and determines an appropriate authentication scheme based on mapping rules that identify those authentication schemes which are sufficient to achieve a given trust level. Preferably, the mapping rules are a function of environment information. In some configurations, mapping rules are implemented as fuzzy sets wherein acceptable authentication schemes are a function of required trust level and environment information. In this way, environment affects the set of authentication schemes sufficient to meet a trust level requirement.</p> <p>Wood at p. 16:</p> <p>Gatekeeper/entry handler component 110 again supplies (18) authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session (e.g., the signed session credentials. authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency the session credentials (and underlying authentication of login credentials) for the trust level required for the requested access. Login credentials should now be sufficient for access to the requested resource and an ALLOW response is supplied (19) by authorization component 140. Gatekeeper/entry handler component 110 proxies the requested access (20, 21) to information resource 191 and streams (22) results back to login component 120. Login component 120, in turn, streams (23) results back to browser 170.</p>
--	--

	Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.
[4] The method according to claim 1, wherein the packet is processed utilizing a provisioning module running on the control device, wherein the provisioning module comprises a first table for interface specific rules, a second table for global rules, and a third table for user specific rules.	<p>The Appendix References disclose and/or render obvious: The method according to claim 1, wherein the packet is processed utilizing a provisioning module running on the control device, wherein the provisioning module comprises a first table for interface specific rules, a second table for global rules, and a third table for user specific rules.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 22: While the invention has been described with reference to various embodiments, it will be understood that these embodiments are illustrative and that the scope of the invention is not limited to them. Many variations, modifications, additions, and improvements are possible. For example, the set of authentication schemes described herein is illustrative and embodiments in accordance with the present invention may include others, including those that may be hereafter developed. If employed, rules mapping trust levels to authentication schemes may be encoded in a variety of ways depending on the particular implementation. In general, such mapping rules may be encoded as static or dynamic table associating trust level to authentication schemes. Alternatively, mapping rules may be encoded as predicates or in other declarative forms. Mapping rules may be encoded in a weighted logic or fuzzy set form. Additionally, particular mappings may depend environment information. Furthermore, evaluation of mapping rules may be performed in a variety of functional components such as an authorization service, a credential gathering service or a gatekeeper. Session continuity may be provided using cryptographically secure session tokens passed as cookies or by other mechanisms.</p> <p>Teraslinna '492 at 6:50-55: The bandwidth parameters associated with the source endpoint identified by the address field are obtained from a look-up table, as indicated by block 42. The bandwidth parameters provide measures of the bandwidth associated with a bundle of provisioned virtual connections emanating from the source endpoint.</p>

	<p>Teraslinna '492 at 7:8-15: In a second embodiment of a method of performing the step indicated by block 54, an intermediate pointer label is obtained from the look-up table in dependence upon the source label. The intermediate pointer label and the destination label are then concatenated. A second table look-up step is performed based upon the concatenated label. This returns the physical switch port address as well as a new outgoing address field.</p> <p>Teraslinna '492 at 8:26-31: In a first method, a table is provisioned in each piece of subscriber equipment by the network manager at network provisioning time. The table in each subscriber equipment contains the address of each other subscriber, and associated with it, the address field to be used to communicate between the subscriber and any of the other subscribers.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[5] The method according to claim 1, wherein the user device is assigned with an Internet Protocol (IP) address in the first network and wherein the at least one user specific rule is associated with the IP address.</p>	<p>The Appendix References disclose and/or render obvious: The method according to claim 1, wherein the user device is assigned with an Internet Protocol (IP) address in the first network and wherein the at least one user specific rule is associated with the IP address.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 7: Of note, in certain embodiments in accordance with the present invention, the mapping of login credential types and authentication mechanisms to trust levels is influenced by environment information such as time of request, source of request, connection speed, and/or client application (e.g., browser) environment information. In this way, even with a static set of mapping rules, the set of credential types and authentication mechanisms suitable to support a given trust level may vary based on environment information. In general, mapping rule dependencies are based on perceived variations in threat characteristics and/or requesting entity capabilities. In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 is the authority on environment information for a particular requesting entity.</p>

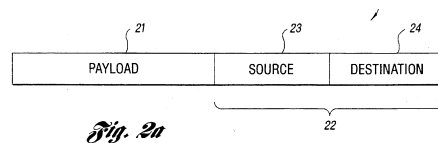
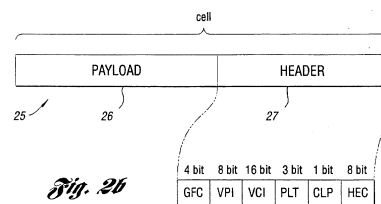
	<p>Wood at p. 11:</p> <p>Entry handler functionality (e.g., in gatekeeper/entry handler component 110) ascertains much of the requesting client's environment information. For example, for dial-up connections, environment information such as line speed and low-level line encryption are ascertained. Additional information, such as source number (e.g., from caller id information or based on a callback configuration), signaling type (e.g., POTS or digital ISDN), etc., may also be obtained. For network connections, similar environment information (e.g., source network and/or node, Virtual Private Network (VPN) low-level encryption, etc.) may be obtained from incoming requests themselves or based on a particular entry point (e.g., a particular router or port). More generally, gatekeeper/entry handler component 110 obtains and/or maintains information such as connect location (if ascertainable), temporal information such as request time/date, session start time/date, etc. (preferably in both the client entity's frame of reference and the security architecture or requested information resource's frame of reference, if location is ascertainable), and client type and/or capabilities (e.g., browser type and Java Development Kit (JDK) level). In some configurations, information on line speed, origination point (e.g., inside or outside of a corporate network), browser type, encryption capability, number of hops, latency, system type, etc. may be gathered. Such information is used in some configurations for mapping particular authentication mechanisms to trust levels and for authorization decisions. Environment information is generally packaged into a data structure that is associated with a client session. Other components of the security architecture may add additional client environment information, such as authentication strength or current trust level.</p> <p>Teraslinna '492 at 8:10-25:</p> <p>Embodiments of the present invention are further illustrated in terms of a virtual private network, which is to be implemented for a plurality of subscribers. First, the network is provisioned with virtual connections between all subscriber endpoints. This consists of writing data into all the fast packet switches associating addresses and bandwidth parameters with all the virtual connections. This is performed by the network managers. When a transmission of information is to be made from a first subscriber to a second subscriber, the first subscriber packetizes the information, and includes the proper address field in a header of each packet. For two-way communications, the second subscriber packetizes its information to be sent to the first subscriber, and includes the proper address field in the header of each packet. The proper address field can be found by the subscriber in the following two ways.</p>
--	---

	<p>Teraslinna '492 at 8:26-31:</p> <p>In a first method, a table is provisioned in each piece of subscriber equipment by the network manager at network provisioning time. The table in each subscriber equipment contains the address of each other subscriber, and associated with it, the address field to be used to communicate between the subscriber and any of the other subscribers.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[6] The method according to claim 5, wherein the user device is associated with a media access control (MAC) address and wherein the at least one user specific rule is further associated with the MAC address.</p>	<p>The Appendix References disclose and/or render obvious: The method according to claim 5, wherein the user device is associated with a media access control (MAC) address and wherein the at least one user specific rule is further associated with the MAC address.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 16:</p> <p>Although the encoding of session tokens using cookies is presently preferred, in part because cookie facilities are widely supported and reasonably well accepted, other facilities may be employed to establish session continuity. For example, alternative URL encodings and/or custom or plug-in support for session identifier receipt, storage and supply may be employed. Also, some configurations may employ lower-level session identifiers, e.g., as provided by a particular connection type such as trusted caller id information or as associated with a low-level line encryption or virtual private network infrastructure. As such facilities are likely to be connection-type specific, it is envisioned that they may be used in conjunction with other session identifier facilities described above, e.g., session tokens encoded in cookies. In one configuration, the unique Ethernet MAC address associated with a network interface card may be employed as a session handle. The MAC address is then used to associate with a particular set of session credentials maintained by a central authority. In general the representation of a session handle can take many forms.</p> <p>Teraslinna '492 at 4:11-21:</p> <p>The address may identify, but need not identify a unique global address of the user at the source and destination endpoints such as a telephone number. In a preferred embodiment, the address identifies a hop or segment of a virtual connection in a network of fast packet switches.</p>

	<p>Translations in a fast packet switch may change the address in the packet for subsequent hops of the same virtual connection. In other words, the address locally identifies a virtual connection, and is uniquely associated with, and therefore indirectly identifies, the source and destination endpoints connected.</p> <p>Teraslinna '492 at 5:1-10: Referring back to FIG. 1, the method further includes a step of enforcing a bandwidth constraint, as indicated by block 12, based upon the source endpoint identified by the address field, wherein the bandwidth constraint is independent of the destination endpoint identified by the address field. In a preferred embodiment wherein the address field contains a source label which identifies the source endpoint and a destination label which identifies the destination endpoint, the bandwidth constraint is enforced based upon the source label and is independent of the destination label.</p> <p>Teraslinna '492 at 5:47-60: The method of FIG. 1 can be employed to provide bandwidth resource management in a communication network wherein a plurality of virtual connections are provisioned from a source endpoint to a plurality of potential destination endpoints, and wherein the desired destination endpoint identified by the address field of the received packet is included in the plurality of potential destination endpoints. It is noted that the address field need not explicitly identify the desired destination endpoint; for example, the address field may identify the virtual connection between the source endpoint and the desired destination endpoint. Here, the desired destination endpoint can be found from a database based upon the virtual connection identified by the address field.</p> <p>Teraslinna '492 at 6:64-67: If the traffic contract is not violated, then a new outgoing address field and a physical switch port address are obtained by the look-up table, as indicated by block 54. Two embodiments of performing this step are described below.</p> <p>Teraslinna '492 at 8:10-25: Embodiments of the present invention are further illustrated in terms of a virtual private network, which is to be implemented for a plurality of subscribers. First, the network is provisioned with virtual connections between all subscriber endpoints. This consists of writing</p>
--	---

data into all the fast packet switches associating addresses and bandwidth parameters with all the virtual connections. This is performed by the network managers. When a transmission of information is to be made from a first subscriber to a second subscriber, the first subscriber packetizes the information, and includes the proper address field in a header of each packet. For two-way communications, the second subscriber packetizes its information to be sent to the first subscriber, and includes the proper address field in the header of each packet. The proper address field can be found by the subscriber in the following two ways.

Teraslinna '492 at Fig. 2a-2b



Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[7] The method according to claim 1, wherein the user specific traffic control rules and the user specific firewall rules are established for the user based on a user profile that includes parameters that determine an extent of the user's access to the second network,

The Appendix References disclose and/or render obvious: The method according to claim 1, wherein the user specific traffic control rules and the user specific firewall rules are established for the user based on a user profile that includes parameters that determine an extent of the user's access to the second network, services available to the user, or a combination thereof.

See, e.g.:

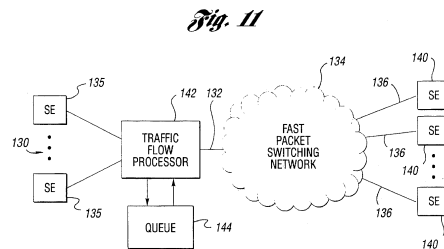
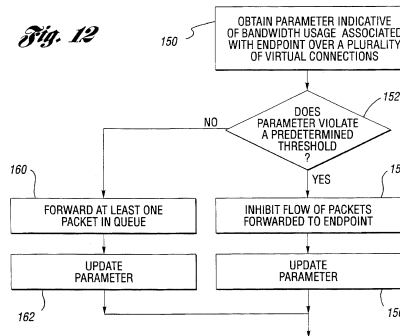
<p>services available to the user, or a combination thereof.</p>	<p>Wood at p. 6-7: Login component 120, operating in conjunction with gatekeeper/entry handler component 110 and other components of the security architecture, provides a single sign-on interface for access to enterprise applications and/or resources. In an exemplary embodiment, security requirements are expressed in terms of trust levels and login component 120 obtains login credentials for an entity requesting access to one of the enterprise applications and/or resources. The login credentials obtained are selected from a set of credential types that, if authenticated, are sufficient to achieve the trust level requirement of an application or information resource to be accessed. Without limitation, typical login credential types and associated authentication mechanisms include those based on passwords, certificates, biometric techniques, smart cards, etc. Other credential types and associated authentication mechanisms are described elsewhere herein.</p> <p>Wood at p. 9: In some embodiments, a trust level encodes the authorization level to which a principal has been authenticated. In such embodiments, the encoded trust level serves as a basis for evaluating whether a principal associated with the session credentials has been authenticated to a sufficient level for access to a requested resource. For example, a trust level of 5 may be sufficient for access to information resources having a trust level requirement of 5 or less. Trust levels offer an attractive decoupling of authorization levels and authentication methods as described elsewhere herein. However, in some embodiments, an authorization encoding may establish that a principal has been authenticated using a particular authentication mechanism. In either case, an authorization (e.g., encoded as a trust level) in a cryptographically secured session credential allows the security architecture to authorize accesses based on prior authentication of a login credential and without involvement of the authentication service.</p> <p>Wood at p. 12-13: Gatekeeper/entry handler component 110 supplies authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session. Preferably, the associated session identifier is cryptographically secured (e.g., as a signed session credential). In some configurations, the signed session credential is obtained from the corresponding session object. In the case of a pre-existing session, the signed session credential may be obtained using a received session token. In any case, authorization component 140 receives (3) the requested resource and session identifiers and responds (4) in accordance</p>
--	---

	<p>with the trust level requirement of the requested resource. In configurations for which sensitivity to a changing environment is desired, environment information may also be supplied to authorization component 140. In an exemplary configuration, authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency of a current trust level. In some configurations, authorization component 140 dynamically calculates a current trust level based on the signed session credentials and environment information. In others, authorization component 140 may base its ALLOW, REDIRECT, or REFUSE response on a "current" trust level previously associated with the signed session credentials. Generally, an access request with sufficient current trust level is ALLOWED and forwarded (14) without further authentication. An authorization request without proper parameters (e.g., without a specified information resource or without a properly secured session credential) may be REFUSED. Authorization requests associated with a client entity that has been blacklisted or for a resource for which the associated client entity cannot be authenticated using any available method to achieve the required trust level may also be REFUSED. For example, a given security policy and associated trust level mappings may dictate a REFUSE response in response to an access request to a sensitive resource (such as financial data) by any client entity (even a browser supplying the digital certificate for the CFO, and therefore presumably operating on behalf of the CEO) if the access request is received over a dial-up line and originates from an unknown number or is received outside of working hours.</p> <p>Wood at p. 14:</p> <p>Generally, mapping rule logic is typically evaluated before a user is challenged to authenticate. Mapping occurs as a function of session environment and particulars of the information resource for which access is requested. By evaluating the minimum trust level required by the target of an access request, a service (e.g., a login service such as provided by login component 120) derives a list of potential authentication methods. The service then checks current session environment against the allowed environment states for each potential authentication method to trim the list further. If there is no particular resource for which access is being requested (e.g., if a user jumps straight to a sign-on page without requesting an access), the service will proceed according to the lowest level of trust available consistent with session environment. Other configurations may employ differing default behaviors.</p>
--	--

	<p>Wood at p. 16: Gatekeeper/entry handler component 110 again supplies (18) authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session (e.g., the signed session credentials. authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency the session credentials (and underlying authentication of login credentials) for the trust level required for the requested access. Login credentials should now be sufficient for access to the requested resource and an ALLOW response is supplied (19) by authorization component 140. Gatekeeper/entry handler component 110 proxies the requested access (20, 21) to information resource 191 and streams (22) results back to login component 120. Login component 120, in turn, streams (23) results back to browser 170.</p> <p>Teraslinna '492 at 5:16-25: The bandwidth constraint is based upon at least one predefined measure of the bandwidth associated with the source identified by the address field. As previously mentioned, such measures of bandwidth include average bit rate and peak bit rate. If the bandwidth constraint for the source endpoint is violated by an incoming packet in accordance with a predetermined traffic contract, then the packet is penalized in a predetermined manner. As described earlier, the violating packet can be discarded or marked with a low priority for loss, for example.</p> <p>Teraslinna '492 at 5:61-6:2: The bandwidth constraint for the source endpoint is enforced by measuring the bandwidth usage associated with the source endpoint over all of the plurality of virtual connections. If the packet causes the violation of a predetermined bandwidth constraint, then the packet is penalized as described earlier. In this context, the plurality of virtual connections which emanate from the source endpoint can be considered as a bundle for the purpose of assigning bandwidth thereto and enforcing bandwidth constraints therefor.</p> <p>Teraslinna '492 at 15:40-49: The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint 132 over at least two of the virtual connections. The at least one parameter indicative of</p>
--	---

	<p>bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.</p> <p>Teraslinna '492 at 15:50-59: The at least one parameter parallels at least one of the bandwidth parameters monitored and enforced by the network 134 in accordance with a predetermined traffic contract. Preferably, the traffic flow processor 142 obtains each of the parameters which are monitored by the network 134 in accordance with the predetermined traffic contract. The at least one parameter may be indicative of an average bit rate or a peak bit rate, for example, associated with the bundle. The at least one parameter may include a leaky bucket parameter as described earlier.</p> <p>Teraslinna '492 at 16:13-21: When the at least one parameter complies with the predetermined traffic contract, i.e., each parameter complies with a corresponding predetermined threshold, the traffic flow processor 142 performs a step of forwarding one of the at least one packet stored in the queue 144 to the endpoint 132. The traffic flow processor 142 further updates each of the at least one parameter when the packet is forwarded to the endpoint 132. Typically, the leaky bucket parameter is updated by performing a step of incrementing the parameter.</p>
--	---

Teraslinna '492 at Fig. 11-12:



Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[8] A method according to claim 1, further comprising determining whether the user has authenticated.

The Appendix References disclose and/or render obvious: A method according to claim 1, further comprising determining whether the user has authenticated.

See, e.g.:

Wood at p. 1:

(57) Abstract: A security architecture has been developed in which a single sign-on is provided. Session credentials are used to maintain continuity of a persistent session across multiple

	<p>accesses to one or more information resources, and in some embodiments, across credential level changes. Session credentials are secured, e.g., as a cryptographically secured session token, such that they may be inspected by a wide variety of entities or applications to verify an authenticated trust level, yet may not be prepared or altered except by a trusted authentication service. Some embodiments of the present invention associate trust level requirements with information resources. Authentication schemes (e.g., those based on passwords, certificates, biometric techniques, smart cards, etc.) are associated with trust levels, and in some embodiments, with environmental parameters. For example, in one configuration, a login service (120) obtains login credentials for an entity (e.g., 170) commensurate with the trust level requirement(s) of an information resource or information resources (e.g., 191, 192, 193) to be accessed and with environment parameters that affect the sufficiency of a given credential type. Once login credentials (e.g., 410) have been obtained for an entity and have been authenticated to a given trust level, session credentials (e.g., 420) are issued and access is granted to information resources for which the trust level is sufficient. Advantageously, by using the session credentials access is granted without the need for further login credentials and authentication. In some configurations, session credentials evidencing an insufficient trust level may be remedied by a session continuity preserving upgrade of login credential.</p> <p>Wood at p. 7:</p> <p>In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 queries authorization component 140 to obtain authorization for access to a particular requested enterprise application or information resource by the requesting entity (e.g., the browser user). If the entity requesting access has not yet been authenticated to the trust level required for the particular access to the particular enterprise application or information resource requested, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that login credentials may be obtained and authenticated to a particular trust level. If, on the other hand, login credentials have already been obtained for the requesting entity and the requesting entity has been authenticated using the obtained credentials such that the required trust level has been achieved, the access will typically be allowed without the need for further login credentials and authentication. In certain circumstances, authorization component 140 may indicate that the access is to be refused. For example, authorization component 140 may be programmed to perform more stringent testing beyond a trust level requirement. In an exemplary enterprise tool configuration, a desired security policy may dictate</p>
--	---

	<p>that a salary tool is accessible only from with a company's internal network. No level of authenticated trust may be sufficient to access such a tool from outside company network. To facilitate implementation of such a security policy, authorization component 140 could refuse access based on environment parameters indicating a session originating outside the company's internal network.</p> <p>Wood at p. 11: Gatekeeper functionality (e.g., in gatekeeper/entry handler component 110) checks whether a session is already associated with the incoming request. Although other techniques are possible, in some configurations in accordance with the present invention, gatekeeper/entry handler component 110 checks for the presence of a session token in the incoming request. Use of session tokens is described in greater detail below; however, in short, a session token may be any data supplied to the client entity for use in uniquely identifying an associated session. In general, preferred session token implementations are cryptographically secured and include facilities, such as expiration or mapping to a particular connection, to limit risk of replay attack and/or spoofing. Some session token implementations may encode session, principal, and/or trust level information. Some session token implementations may employ cookies, URL encoding, or other similar techniques for binding to incoming requests.</p> <p>Wood at p. 12: Referring again to FIG. 1, if a session token is present in the incoming request, gatekeeper/entry handler component 110 resolves the token to a session object. Alternatively, if no session token is present or a session token is invalid, gatekeeper/entry handler component 110 establishes a new session (2). In an exemplary configuration in accordance with FIG. 1, session management component 160 allocates a new session and supplies associated default session credentials (2) based on the requested information resource and environment information. Note that a session is created irrespective of authentication status or identity, although some implementations may refuse to even allocate a session based on particular information resource requests and/or environment information. Given a session object, which may be resolved from a received session token or newly allocated, gatekeeper/entry handler component 110 interacts (3, 4) with authorization component 140 to determine whether the requested access is authorized. For some requested accesses and security policies (e.g., anonymous ftp access to certain archives), even a</p>
--	---

	<p>session without authenticated login credentials (trust level=0) may be authorized. For others, a more substantial trust level may be required.</p> <p>Wood at p. 12-13:</p> <p>Gatekeeper/entry handler component 110 supplies authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session. Preferably, the associated session identifier is cryptographically secured (e.g., as a signed session credential). In some configurations, the signed session credential is obtained from the corresponding session object. In the case of a pre-existing session, the signed session credential may be obtained using a received session token. In any case, authorization component 140 receives (3) the requested resource and session identifiers and responds (4) in accordance with the trust level requirement of the requested resource. In configurations for which sensitivity to a changing environment is desired, environment information may also be supplied to authorization component 140. In an exemplary configuration, authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency of a current trust level. In some configurations, authorization component 140 dynamically calculates a current trust level based on the signed session credentials and environment information. In others, authorization component 140 may base its ALLOW, REDIRECT, or REFUSE response on a "current" trust level previously associated with the signed session credentials. Generally, an access request with sufficient current trust level is ALLOWED and forwarded (14) without further authentication. An authorization request without proper parameters (e.g., without a specified information resource or without a properly secured session credential) may be REFUSED. Authorization requests associated with a client entity that has been blacklisted or for a resource for which the associated client entity cannot be authenticated using any available method to achieve the required trust level may also be REFUSED. For example, a given security policy and associated trust level mappings may dictate a REFUSE response in response to an access request to a sensitive resource (such as financial data) by any client entity (even a browser supplying the digital certificate for the CFO, and therefore presumably operating on behalf of the CEO) if the access request is received over a dial-up line and originates from an unknown number or is received outside of working hours.</p>
--	---

	<p>Wood at p. 14-15:</p> <p>In some configurations, login component 120 queries (7) authorization component 140 to identify the set of authentication schemes that meet or exceed the required trust level given a current environment. In other configurations, the mapping is performed by login component 120. In either case, login component 120 supplies (9) information to browser 170 to allow the user to select from the suitable authentication schemes and to provide an associated login credential. In some configurations, login component 120 supplies browser 170 with a login page (e.g., HTML) that prompts the user for an application specific user ID and a choice of authentication schemes. Interactions with browser 170 depend on the set of credential types that, if authenticated, would be sufficient to meet the trust level requirement for access to the requested resource. For example, if more than one type of credential would be sufficient, login component 120 may interactively allow a user to select from amongst the credential types (e.g., using any HTML-based dialog). Once a particular credential type has been selected, login component 120 interacts with browser 170 to obtain an instance of the login credential to establish the identity of the browser user.</p> <p>Wood at p. 16-17:</p> <p>Referring again to FIG. 1, subsequent access requests (e.g., access request 1A) include a previously assigned session token. As described above, gatekeeper/entry handler component 110 uses the session token, if provided to resolve a session object containing session credentials, and to determine whether previously authenticated credentials are sufficient for the requested access. As described above, authorization component 140 may be queried using session credentials and an identifier for the requested resource to determine sufficiency of previously authenticated credentials. In some configurations, sufficiency is determined using trust level mappings as described above. Depending on the information resource to which access is requested, and in some configurations depending on current session environment information, access request 1A may or may not have associated previously authenticated credentials sufficient to support the requested access. In the case of an access request 1A having a trust level requirement commensurate with previously obtained and authenticated credentials (i.e., an access request for which no additional credentials need be obtained via login component 120), the access request is proxied (20) and results (21) are streamed directly (23A) back to browser 170. In some configurations, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated</p>
--	---

	<p>session token, if supplied, resolves to the same session object as the session token replaced. For example, in some configurations, both session tokens encode a same session handle, although other aspects associated with session token security such as expiration may be updated. In other configurations, results (21) are streamed (23A) back to browser 170 without an updated session token. In such configurations, the previously supplied session token remains valid until expired or otherwise invalidated. Some variations may employ a session token refresh interval.</p> <p>Wood at p. 17: Depending on the information resource to which access is requested, previously obtained and authenticated login credentials may be insufficient for the trust level requirement associated with requested access 1A. As before, authorization component 140 supplies gatekeeper/entry handler component 110 with an ALLOW, REDIRECT or REFUSE response based on the trust level accorded based on the previously obtained and authenticated login credentials and on the trust level requirement associated with requested access 1A. Advantageously, authorization of individual access requests is streamlined by the encoding of trust level in a cryptographically secured session credential or token. In the case of insufficient credentials, a REDIRECT response is supplied and gatekeeper/entry handler component 110 again redirects (5) browser 170 to login component 120. Additional login credentials are obtained as described above with reference to initial credentials. Upon successful authentication, access request is proxied (20) and results (21) are streamed (23A) back to browser 170.</p> <p>Wood at p. 18: FIG. 2 illustrates operation of an exemplary security architecture providing a single sign-on facility with trust level mapping to authentication requirements. As before, an access request is received (201) from a client entity. If the request does not contain a session identifier (e.g., a session key or token) or if the request can otherwise be reliably associated with a session maintained by the security architecture, a new session is created (202). A trust level requirement is determined for access to the requested resource in the context of the requesting session environment. In some configurations, as described above, the determination is performed by an authorization service such as authorization component 140. Given a trust level requirement, current session credentials are evaluated (203) in the context of session environment information to determine whether the previously supplied login credentials are sufficient to achieve the required trust level. In one advantageous realization of session credentials and tokens, a</p>
--	---

	<p>cryptographically secured encoding of trust level allows authorization to be confirmed without involvement of an authentication service (e.g., with reauthentication of login credentials). In the case of a newly created (202) session, the authorization evidenced by session credentials will typically be insufficient, although some security policies may allow anonymous access to certain resources.</p> <p>Wood at p. 18: As described elsewhere herein, sufficiency of current session credentials is determined using mapping rules that, in some realizations, include environment information. In general, current session credentials may be insufficient (1) because the identity of the requesting client has not yet been authenticated (e.g., in a first access situation), (2) because of a higher trust level requirement for the requested access, or (3) because of a change in mapping rules or environment that causes a previously sufficient credential no longer be sufficient for a particular trust level. Whatever the reason for the insufficiency, a request corresponding to a session and client entity that is insufficiently authenticated, and that is therefore not authorized, is passed to a facility for obtaining credentials of a type that, if authenticated, will support the required trust level.</p> <p>Wood at p. 19: Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information. Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive 20 process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents back to the credential gathering service. Others (e.g., certificates) may be obtained for the client</p>
--	--

	<p>entity from an agent or authority. For example, a personal digital certificate (such as those issued by VerisignTM, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select 25 from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.</p> <p>Wood at p. 21:</p> <p>If an exception is thrown due to insufficient authorization, e.g., if the signed session credentials do not indicate that the requesting entity is sufficiently authorized to access order management service 312, a login credential gathering process is initiated. Based on the required trust level and on rules that encode the sufficiency of authentication schemes, a login credential is obtained from user 301 and/or browser 302. The obtained login credential is of a type that, if authenticated, is sufficient to meet the trust level requirement for access to order management service 312. Aspects of an exemplary credential gathering process are described in greater detail above. However, as an example, FIG. 3 illustrates the obtaining of a username/password pair. Once login credentials are obtained, they are passed to central security framework 304 (as described above) for authentication by central authentication service 307 so that session credentials can be obtained, the requested access can be authorized, and the order processing initiated. Signed session credentials, typically embodied as a cryptographically secured session token that may be stored as a cookie, are passed back to browser 302 with results of the requested access. In this way, subsequent access requests are identified as part of a session and authorization may be quickly confirmed without unnecessary re-authentication.</p>
--	---

[illegible]

FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention.

FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention. As shown, the operation and function of the components of the architecture for providing the integrated, distributed shared services system illustrated in FIG. 4 is further discussed. Accordingly, reference to the components shown in FIG. 4 will be made in conjunction with the operation and function illustrated in FIG. 5.

Lerner '776 at [0063]:

First, at step 501 the user terminal client web browser 412 submits an HTTP request with a cookie to a web application 401 such that the URL of the desired resource (e.g., application 401) available in the central server domain and the cookie is transmitted to the application 401 . Having received the HTTP request and the cookie, at step 502 the application corresponding to the HTTP request (e.g., application 401) transmits the cookie to the application interface library 402 . The application interface library 402 at step 503 then performs user authentication processing. In particular, the application interface library 402 transmits the cookie received from the client web browser 412 to the cookie access library 405 which is configured to read the cookie data at step 504 . In the event that the transmitted cookie is encrypted, the cookie access library 405 transmits the encrypted cookie to the encryption/ decryption engine 403 which, at step 505 decrypts the encrypted cookie and passes the decrypted cookie back to the cookie access library 405 . The cookie access library 405 then parses the decrypted cookie at step 506 into cookie data and accordingly, determines whether the user ID is matched or not by returning a confirm authentication or a null match command to the application interface library 402 . With a matched user ID, the application interface library 402 returns the result to the web application 401 which then delivers the application content at step 507 for display at the client browser 412 at step 508 . On the other hand, if the authentication process is unsuccessful, a null match command is returned to the web application 401 which then delivers redirect information with a HTTP redirect to the client web browser 412 at step 509 . The client web browser 412 then submits the HTTP redirect request to the login page of the application at step 510 , and the login page URL and the cookie is sent to the login page at the client web browser 412 at step 511 where the user is prompted to input login information to access the desired application.

Lerner '776 at [0064]:

FIG. 6 illustrates a flowchart for a user event processing procedure to access a desired application in accordance with one embodiment of the present invention. As shown, when a user event occurs such as when a user/client requests access to a particular application or resource in the central server domain, at step 601 the user at the user terminal web browser 412 submits an HTTP request corresponding to the requested application or resource, transmitting the corresponding URL and the cookie file. At step 602 , the corresponding web application 401 receives the HTTP request and the cookie file, and in turn, passes the event name and the

	<p>received cookie to the application interface library 402 which posts the corresponding event at step 603 . In other words, at step 603 , the requested application or data store from the application interface library 402 places the message signifying the user event onto either a browser cookie or a message queue for eventual delivery at step 607 to another (or subsequent) data store or application. Again, the application, event and the user ID in the form of a message can be optionally encrypted at step 608 by the encryption engine 409 when they are placed onto the message queue before being transmitted from the message queuing middleware 404 of the application or data store resident locally at the central server or at a third party vendor site within the central server domain.</p> <p>Lerner '776 at [0070]: Furthermore, the architecture in accordance with the present invention provides a vehicle for different user registration and authentication components to operate together. This feature is invaluable in integrating public web sites and secure site registration authentication and registration, as well as in integrating any outsourced registration/authentication with in house components. With respect to the web applications, any application that can call a library and be location in the central server domain will be able to use the shared services for its registration and authentication. Indeed, the cookie-based authentication allows a user to log in once and move from application to application without the need for re-authentication. Additionally, user behavior and user provided information can be passed to a central user profile repository using the message passing portion of the architecture. The relevant profile can then be placed in the user's browser cookie at the time of user login process. Thus, each application can read the cookie and use it to shape banners, prompts and other content based on the user behavior and provided information.</p>
--	---

Lerner '776 at Fig. 5:

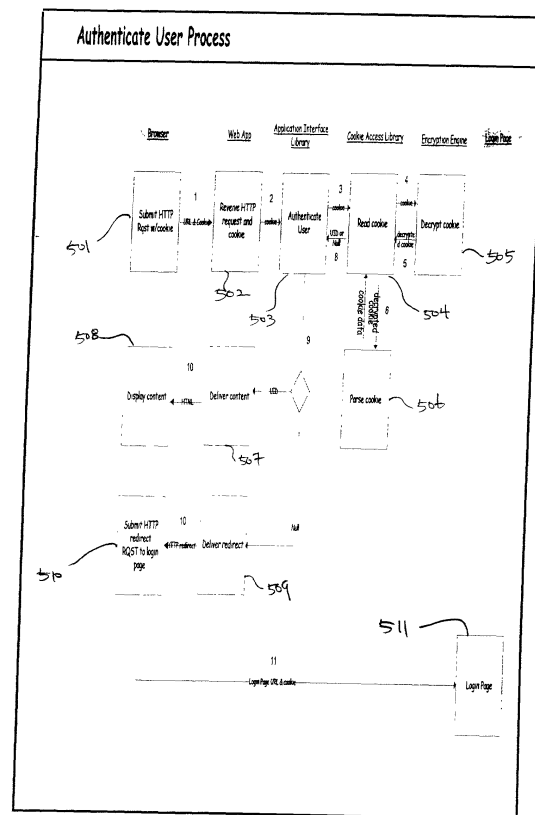


FIGURE 5

Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[9] A method according to claim 8, wherein the user has not authenticated, further comprising:

The Appendix References disclose and/or render obvious: A method according to claim 8, wherein the user has not authenticated.

See, e.g.:

	<p>Wood at p. 1:</p> <p>(57) Abstract: A security architecture has been developed in which a single sign-on is provided. Session credentials are used to maintain continuity of a persistent session across multiple accesses to one or more information resources, and in some embodiments, across credential level changes. Session credentials are secured, e.g., as a cryptographically secured session token, such that they may be inspected by a wide variety of entities or applications to verify an authenticated trust level, yet may not be prepared or altered except by a trusted authentication service. Some embodiments of the present invention associate trust level requirements with information resources. Authentication schemes (e.g., those based on passwords, certificates, biometric techniques, smart cards, etc.) are associated with trust levels, and in some embodiments, with environmental parameters. For example, in one configuration, a login service (120) obtains login credentials for an entity (e.g., 170) commensurate with the trust level requirement(s) of an information resource or information resources (e.g., 191, 192, 193) to be accessed and with environment parameters that affect the sufficiency of a given credential type. Once login credentials (e.g., 410) have been obtained for an entity and have been authenticated to a given trust level, session credentials (e.g., 420) are issued and access is granted to information resources for which the trust level is sufficient. Advantageously, by using the session credentials access is granted without the need for further login credentials and authentication. In some configurations, session credentials evidencing an insufficient trust level may be remedied by a session continuity preserving upgrade of login credential.</p> <p>Wood at p. 7:</p> <p>In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 queries authorization component 140 to obtain authorization for access to a particular requested enterprise application or information resource by the requesting entity (e.g., the browser user). If the entity requesting access has not yet been authenticated to the trust level required for the particular access to the particular enterprise application or information resource requested, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that login credentials may be obtained and authenticated to a particular trust level. If, on the other hand, login credentials have already been obtained for the requesting entity and the requesting entity has been authenticated using the obtained credentials such that the required trust level has been achieved, the access will typically be allowed without the need for further login credentials and authentication. In certain circumstances, authorization</p>
--	--

	<p>component 140 may indicate that the access is to be refused. For example, authorization component 140 may be programmed to perform more stringent testing beyond a trust level requirement. In an exemplary enterprise tool configuration, a desired security policy may dictate that a salary tool is accessible only from with a company's internal network. No level of authenticated trust may be sufficient to access such a tool from outside company network. To facilitate implementation of such a security policy, authorization component 140 could refuse access based on environment parameters indicating a session originating outside the company's internal network.</p> <p>Wood at p. 8: In certain embodiments in accordance with the present invention, a credential upgrade facility is provided. In response to an access request from an entity for which login credentials have already been obtained and authenticated, but for which the obtained and authenticated login credentials are insufficient for the trust level associated with the requested access, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that sufficient login credentials may be obtained and authenticated to the required trust level. Of note, credential upgrade facilities in accordance with certain embodiments of the present invention allow upgrade without loss of session continuity.</p> <p>Wood at p. 12-13: Gatekeeper/entry handler component 110 supplies authorization component 140 with an identifier for the requested resource (e.g., the requested URL) and an identifier for the associated session. Preferably, the associated session identifier is cryptographically secured (e.g., as a signed session credential). In some configurations, the signed session credential is obtained from the corresponding session object. In the case of a pre-existing session, the signed session credential may be obtained using a received session token. In any case, authorization component 140 receives (3) the requested resource and session identifiers and responds (4) in accordance with the trust level requirement of the requested resource. In configurations for which sensitivity to a changing environment is desired, environment information may also be supplied to authorization component 140. In an exemplary configuration, authorization component 140 responds with an ALLOW, REDIRECT, or REFUSE response based on the sufficiency of a current trust level. In some configurations, authorization component 140 dynamically calculates a current trust level based on the signed session credentials and environment information. In</p>
--	---

	<p>others, authorization component 140 may base its ALLOW, REDIRECT, or REFUSE response on a "current" trust level previously associated with the signed session credentials. Generally, an access request with sufficient current trust level is ALLOWED and forwarded (14) without further authentication. An authorization request without proper parameters (e.g., without a specified information resource or without a properly secured session credential) may be REFUSED. Authorization requests associated with a client entity that has been blacklisted or for a resource for which the associated client entity cannot be authenticated using any available method to achieve the required trust level may also be REFUSED. For example, a given security policy and associated trust level mappings may dictate a REFUSE response in response to an access request to a sensitive resource (such as financial data) by any client entity (even a browser supplying the digital certificate for the CFO, and therefore presumably operating on behalf of the CEO) if the access request is received over a dial-up line and originates from an unknown number or is received outside of working hours.</p> <p>Wood at p. 13: A REDIRECT response is used to forward the access request to login component 120 so that sufficient login credentials may be obtained and authenticated to achieve the required trust level for the requested access. Note that in some configurations, both initial login credentialing and credential upgrade are provided using the REDIRECT facility. In response to a REDIRECT response (4), gatekeeper/entry handler component 110 redirects (5) browser 170 to login component 120. In one configuration, gatekeeper/entry handler component 110 issues a client-side redirect via HTTP location directive to forward the request to login component 120. Parameters such as required trust level, requested URL and credential passing method can be encoded in the redirect URL and supplied (6) by browser 170 to login component 120. Alternatively, some parameters can be passed (5A) directly (e.g., through a HttpSession object) although a stateless configuration is preferred.</p> <p>Wood at p. 13-14: A session token is passed to browser 170 in conjunction with the redirect (5) to login component 120. Based on the description herein, persons of ordinary skill in the art will appreciate a number of suitable mechanisms for passing the session token, including those based on cookies and URL encoding. Preferably, mechanisms employed are based on facilities provided by commercially available browsers (e.g., in accordance with HTML 3.x, 4.x or other de-facto standards),</p>
--	---

although customized or plug-in facilities for receiving and supplying session token may be employed. In one suitable configuration, the session token is cryptographically secured and encoded in a cookie placed at browser 170 using a set cookie directive embedded in the redirect. Other configurations may use a less persistent session identification method, such as passing an identifier or session token in the redirect URL without storage at browser 170. Still other configurations may transmit a session token, a session credential, or identifier such as a session handle for storage in a medium such as a smart card. In configurations providing credential upgrade, persistent session identification methods are generally preferred, even for a not yet authenticated client entity, for consistency of approach. Note that although the identity of the client entity may not be authenticated to a sufficient level of trust, the redirected request includes a session token that at least identifies the session. Other configurations may omit the binding of session tokens to sessions of not yet authenticated client entities, though with some increase in complexity of login component 120.

Wood at p. 14-15:

In some configurations, login component 120 queries (7) authorization component 140 to identify the set of authentication schemes that meet or exceed the required trust level given a current environment. In other configurations, the mapping is performed by login component 120. In either case, login component 120 supplies (9) information to browser 170 to allow the user to select from the suitable authentication schemes and to provide an associated login credential. In some configurations, login component 120 supplies browser 170 with a login page (e.g., HTML) that prompts the user for an application specific user ID and a choice of authentication schemes. Interactions with browser 170 depend on the set of credential types that, if authenticated, would be sufficient to meet the trust level requirement for access to the requested resource. For example, if more than one type of credential would be sufficient, login component 120 may interactively allow a user to select from amongst the credential types (e.g., using any HTML-based dialog). Once a particular credential type has been selected, login component 120 interacts with browser 170 to obtain an instance of the login credential to establish the identity of the browser user.

Wood at p. 16-17:

Referring again to FIG. 1, subsequent access requests (e.g., access request 1A) include a previously assigned session token. As described above, gatekeeper/entry handler component 110

uses the session token, if provided to resolve a session object containing session credentials, and to determine whether previously authenticated credentials are sufficient for the requested access. As described above, authorization component 140 may be queried using session credentials and an identifier for the requested resource to determine sufficiency of previously authenticated credentials. In some configurations, sufficiency is determined using trust level mappings as described above. Depending on the information resource to which access is requested, and in some configurations depending on current session environment information, access request 1A may or may not have associated previously authenticated credentials sufficient to support the requested access. In the case of an access request 1A having a trust level requirement commensurate with previously obtained and authenticated credentials (i.e., an access request for which no additional credentials need be obtained via login component 120), the access request is proxied (20) and results (21) are streamed directly (23A) back to browser 170. In some configurations, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated session token, if supplied, resolves to the same session object as the session token replaced. For example, in some configurations, both session tokens encode a same session handle, although other aspects associated with session token security such as expiration may be updated. In other configurations, results (21) are streamed (23A) back to browser 170 without an updated session token. In such configurations, the previously supplied session token remains valid until expired or otherwise invalidated. Some variations may employ a session token refresh interval.

Wood at p. 17:

Depending on the information resource to which access is requested, previously obtained and authenticated login credentials may be insufficient for the trust level requirement associated with requested access 1A. As before, authorization component 140 supplies gatekeeper/entry handler component 110 with an ALLOW, REDIRECT or REFUSE response based on the trust level accorded based on the previously obtained and authenticated login credentials and on the trust level requirement associated with requested access 1A. Advantageously, authorization of individual access requests is streamlined by the encoding of trust level in a cryptographically secured session credential or token. In the case of insufficient credentials, a REDIRECT response is supplied and gatekeeper/entry handler component 110 again redirects (5) browser 170 to login component 120. Additional login credentials are obtained as described above with reference to

	<p>initial credentials. Upon successful authentication, access request is proxied (20) and results (21) are streamed (23A) back to browser 170.</p> <p>Wood at p. 18: FIG. 2 illustrates operation of an exemplary security architecture providing a single sign-on facility with trust level mapping to authentication requirements. As before, an access request is received (201) from a client entity. If the request does not contain a session identifier (e.g., a session key or token) or if the request can otherwise be reliably associated with a session maintained by the security architecture, a new session is created (202). A trust level requirement is determined for access to the requested resource in the context of the requesting session environment. In some configurations, as described above, the determination is performed by an authorization service such as authorization component 140. Given a trust level requirement, current session credentials are evaluated (203) in the context of session environment information to determine whether the previously supplied login credentials are sufficient to achieve the required trust level. In one advantageous realization of session credentials and tokens, a cryptographically secured encoding of trust level allows authorization to be confirmed without involvement of an authentication service (e.g., with reauthentication of login credentials). In the case of a newly created (202) session, the authorization evidenced by session credentials will typically be insufficient, although some security policies may allow anonymous access to certain resources.</p> <p>Wood at p. 18: As described elsewhere herein, sufficiency of current session credentials is determined using mapping rules that, in some realizations, include environment information. In general, current session credentials may be insufficient (1) because the identity of the requesting client has not yet been authenticated (e.g., in a first access situation), (2) because of a higher trust level requirement for the requested access, or (3) because of a change in mapping rules or environment that causes a previously sufficient credential no longer be sufficient for a particular trust level. Whatever the reason for the insufficiency, a request corresponding to a session and client entity that is insufficiently authenticated, and that is therefore not authorized, is passed to a facility for obtaining credentials of a type that, if authenticated, will support the required trust level.</p>
--	---

	<p>Wood at p. 19:</p> <p>Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information. Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents back to the credential gathering service. Others (e.g., certificates) may be obtained for the client entity from an agent or authority. For example, a personal digital certificate (such as those issued by Verisign™, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.</p> <p>Wood at p. 21:</p> <p>If an exception is thrown due to insufficient authorization, e.g., if the signed session credentials do not indicate that the requesting entity is sufficiently authorized to access order management service 312, a login credential gathering process is initiated. Based on the required trust level and on rules that encode the sufficiency of authentication schemes, a login credential is obtained from user 301 and/or browser 302. The obtained login credential is of a type that, if authenticated, is sufficient to meet the trust level requirement for access to order management</p>
--	--

	<p>Lerner '776 at [0023]: FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention.</p> <p>Lerner '776 at [0062]: FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention. As shown, the operation and function of the components of the architecture for providing the integrated, distributed shared services system illustrated in FIG. 4 is further discussed. Accordingly, reference to the components shown in FIG. 4 will be made in conjunction with the operation and function illustrated in FIG. 5.</p> <p>Lerner '776 at [0063]: First, at step 501 the user terminal client web browser 412 submits an HTTP request with a cookie to a web application 401 such that the URL of the desired resource (e.g., application 401) available in the central server domain and the cookie is transmitted to the application 401 . Having received the HTTP request and the cookie, at step 502 the application corresponding to the HTTP request (e.g., application 401) transmits the cookie to the application interface library 402 . The application interface library 402 at step 503 then performs user authentication processing. In particular, the application interface library 402 transmits the cookie received from the client web browser 412 to the cookie access library 405 which is configured to read the cookie data at step 504 . In the event that the transmitted cookie is encrypted, the cookie access library 405 transmits the encrypted cookie to the encryption/ decryption engine 403 which, at step 505 decrypts the encrypted cookie and passes the decrypted cookie back to the cookie access library 405 . The cookie access library 405 then parses the decrypted cookie at step 506 into cookie data and accordingly, determines whether the user ID is matched or not by returning a confirm authentication or a null match command to the application interface library 402 . With a matched user ID, the application interface library 402 returns the result to the web application 401 which then delivers the application content at step 507 for display at the client browser 412 at step 508 . On the other hand, if the authentication process is unsuccessful, a null match command is returned to the web application 401 which then delivers redirect information with a HTTP redirect to the client web browser 412 at step 509 . The client web browser 412 then submits the HTTP redirect request to the login page of the application at step 510 , and the login page URL and the cookie is sent to the login page at the client web browser</p>
--	---

412 at step 511 where the user is prompted to input login information to access the desired application.

Lerner '776 at [0064]:

FIG. 6 illustrates a flowchart for a user event processing procedure to access a desired application in accordance with one embodiment of the present invention. As shown, when a user event occurs such as when a user/client requests access to a particular application or resource in the central server domain, at step 601 the user at the user terminal web browser 412 submits an HTTP request corresponding to the requested application or resource, transmitting the corresponding URL and the cookie file. At step 602, the corresponding web application 401 receives the HTTP request and the cookie file, and in turn, passes the event name and the received cookie to the application interface library 402 which posts the corresponding event at step 603. In other words, at step 603, the requested application or data store from the application interface library 402 places the message signifying the user event onto either a browser cookie or a message queue for eventual delivery at step 607 to another (or subsequent) data store or application. Again, the application, event and the user ID in the form of a message can be optionally encrypted at step 608 by the encryption engine 409 when they are placed onto the message queue before being transmitted from the message queuing middleware 404 of the application or data store resident locally at the central server or at a third party vendor site within the central server domain.

Lerner '776 at [0070]:

Furthermore, the architecture in accordance with the present invention provides a vehicle for different user registration and authentication components to operate together. This feature is invaluable in integrating public web sites and secure site registration authentication and registration, as well as in integrating any outsourced registration/authentication with in house components. With respect to the web applications, any application that can call a library and be located in the central server domain will be able to use the shared services for its registration and authentication. Indeed, the cookie-based authentication allows a user to log in once and move from application to application without the need for re-authentication. Additionally, user behavior and user provided information can be passed to a central user profile repository using the message passing portion of the architecture. The relevant profile can then be placed in the user's browser cookie at the time of user login process. Thus, each application can read the

cookie and use it to shape banners, prompts and other content based on the user behavior and provided information.

Lerner '776 at Fig. 5:

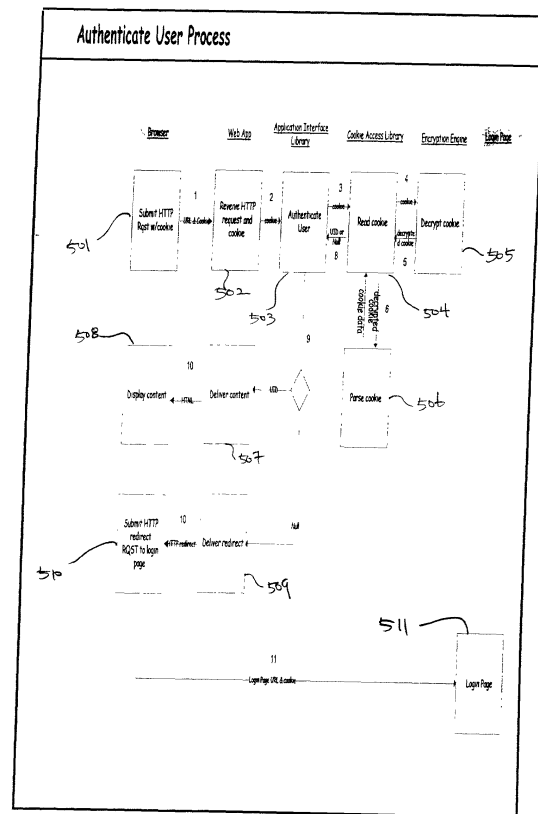


FIGURE 5

Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[9.1] redirecting the user to a login web page that requests user credentials; and

The Appendix References disclose and/or render obvious: redirecting the user to a login web page that requests user credentials.

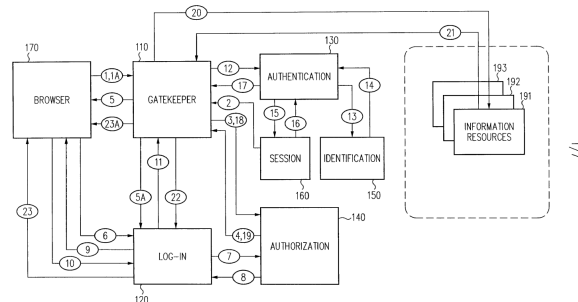
	<p><i>See, e.g.:</i></p> <p>Wood at p. 7:</p> <p>In some embodiments in accordance with the present invention, gatekeeper/entry handler component 110 queries authorization component 140 to obtain authorization for access to a particular requested enterprise application or information resource by the requesting entity (e.g., the browser user). If the entity requesting access has not yet been authenticated to the trust level required for the particular access to the particular enterprise application or information resource requested, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that login credentials may be obtained and authenticated to a particular trust level. If, on the other hand, login credentials have already been obtained for the requesting entity and the requesting entity has been authenticated using the obtained credentials such that the required trust level has been achieved, the access will typically be allowed without the need for further login credentials and authentication. In certain circumstances, authorization component 140 may indicate that the access is to be refused. For example, authorization component 140 may be programmed to perform more stringent testing beyond a trust level requirement. In an exemplary enterprise tool configuration, a desired security policy may dictate that a salary tool is accessible only from with a company's internal network. No level of authenticated trust may be sufficient to access such a tool from outside company network. To facilitate implementation of such a security policy, authorization component 140 could refuse access based on environment parameters indicating a session originating outside the company's internal network.</p> <p>Wood at p. 8:</p> <p>In certain embodiments in accordance with the present invention, a credential upgrade facility is provided. In response to an access request from an entity for which login credentials have already been obtained and authenticated, but for which the obtained and authenticated login credentials are insufficient for the trust level associated with the requested access, authorization component 140 may indicate that the access request is to be redirected to login component 120 so that sufficient login credentials may be obtained and authenticated to the required trust level. Of note, credential upgrade facilities in accordance with certain embodiments of the present invention allow upgrade without loss of session continuity.</p>
--	---

	<p>Wood at p. 13:</p> <p>A REDIRECT response is used to forward the access request to login component 120 so that sufficient login credentials may be obtained and authenticated to achieve the required trust level for the requested access. Note that in some configurations, both initial login credentialing and credential upgrade are provided using the REDIRECT facility. In response to a REDIRECT response (4), gatekeeper/entry handler component 110 redirects (5) browser 170 to login component 120. In one configuration, gatekeeper/entry handler component 110 issues a client-side redirect via HTTP location directive to forward the request to login component 120. Parameters such as required trust level, requested URL and credential passing method can be encoded in the redirect URL and supplied (6) by browser 170 to login component 120. Alternatively, some parameters can be passed (5A) directly (e.g., through a HttpSession object) although a stateless configuration is preferred.</p> <p>Wood at p. 14:</p> <p>Browser 170 sends (6) login component 120 a new access request using the URL specified in the redirect from gatekeeper/entry handler component 110. In configurations employing cookies as a medium for passing session tokens, the new access request will include the cookie and therefore the session token. Note that in configurations in which the security architecture controls access to resources in several domains, care should be exercised to select a tag or tags for the cookie such that it will be provided through normal operation of the browser in subsequent accesses to any of the several domains. Persons of ordinary skill in the art will appreciate suitable tagging techniques, including the use of multiple cookies. Login component 120 receives the access request and determines an appropriate authentication scheme based on mapping rules that identify those authentication schemes which are sufficient to achieve a given trust level. Preferably, the mapping rules are a function of environment information. In some configurations, mapping rules are implemented as fuzzy sets wherein acceptable authentication schemes are a function of required trust level and environment information. In this way, environment affects the set of authentication schemes sufficient to meet a trust level requirement.</p> <p>Wood at p. 14-15:</p> <p>In some configurations, login component 120 queries (7) authorization component 140 to identify the set of authentication schemes that meet or exceed the required trust level given a current environment. In other configurations, the mapping is performed by login component 120.</p>
--	---

	<p>In either case, login component 120 supplies (9) information to browser 170 to allow the user to select from the suitable authentication schemes and to provide an associated login credential. In some configurations, login component 120 supplies browser 170 with a login page (e.g., HTML) that prompts the user for an application specific user ID and a choice of authentication schemes. Interactions with browser 170 depend on the set of credential types that, if authenticated, would be sufficient to meet the trust level requirement for access to the requested resource. For example, if more than one type of credential would be sufficient, login component 120 may interactively allow a user to select from amongst the credential types (e.g., using any HTML-based dialog). Once a particular credential type has been selected, login component 120 interacts with browser 170 to obtain an instance of the login credential to establish the identity of the browser user.</p> <p>Wood at p. 17: Preferably, gatekeeper/entry handler component 110 supplies an updated session token using a set cookie directive encoded with the results streamed (23A) back to browser 170. An updated session token, if supplied, resolves to the same session object as the session token replaced. As a result, session state (including e.g., identity mappings, authorizations, roles, permissions, environmental variables, etc.) is maintained through the credential level change. However, in the case of a credential upgrade, the session object now encodes a login credential successfully authenticated to achieve a higher trust level. In one advantageous configuration, the achieved (higher) trust level is encoded in a cryptographically secured session token representation as a cookie streamed (23A) back to browser 170 with results (21).</p> <p>Wood at p. 19: Referring again to FIG. 2, a request corresponding to a session not authorized for a requested access 10 is redirected (206) to a credential gathering service (e.g., login component 120). The credential gathering service receives (207) the redirected access and obtains (208) a trust level requirement for the requested access. In some configurations, the trust level requirement may be passed with the redirected access or otherwise associated with the redirected access, in others the trust level requirement may be re-obtained from an authorization service such as authorization component 140. A trust level requirement is mapped (209) to at least one authentication scheme sufficient to achieve the requirement based on current trust level mappings and, if employed in the mapping rules, based on current environment information.</p>
--	---

Assuming that at least one authentication scheme is available that, if successful, will support the required trust level, login credentials of that type are obtained (210) for the entity and authenticated (211). Some credential types (e.g., username/password pairs, onetime passwords, enigma challenge, etc) may be obtained through an interactive 20 process in which a principal (e.g., a human user) supplies the credential(s) entry in an HTML form and POSTs form contents back to the credential gathering service. Others (e.g., certificates) may be obtained for the client entity from an agent or authority. For example, a personal digital certificate (such as those issued by VerisignTM, Thwate, Entrust or other certificate authority) may be obtained from a browser using an HTTP certificate request. In some configurations, a choice of credential types may be provided and user may select 25 from a set of credential types sufficient for the requested access. Once appropriate login credentials have been obtained and authenticated, the session corresponding to the requested access is updated (213) to reflect the new authenticated session credentials. The now sufficiently authorized request is proxied (204) and results are streamed back (205) to the requesting client entity. Updated or refreshed session credentials are typically supplied as a session token (e.g., a cookie) encoded with the streamed results.

Wood at Fig. 2:



Lerner '776 at [0009]:

One disadvantage to the outsourcing approach, however, is that in many cases, the users are required to provide login information each time the user logs onto a third party vendor web site. For example, an account holder with C. Schwab & Co., the assignee of the present application, may wish to enroll in an online interactive education courses that are offered through Schwab.com web site but provided by a third party vendor. After initially logging onto schwab.com's web site, the account holder clicks on a hypertext link in schwab.com's web site,

	<p>and the account holder's web browser is directed to the web site of the third party vendor providing the online interactive education courses. Here, the Schwab account holder must log in once again to register or to resume the course from where it was left of during the last session.</p> <p>Lerner '776 at [0012]: As discussed above, irrespective of the type of third party vendors and the application or information provided, a user or an account holder is required to provide login information each time the user wishes to access the third party vendor application from a central server domain such as schwab.com.</p> <p>Lerner '776 at [0023]: FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention.</p> <p>Lerner '776 at [0062]: FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention. As shown, the operation and function of the components of the architecture for providing the integrated, distributed shared services system illustrated in FIG. 4 is further discussed. Accordingly, reference to the components shown in FIG. 4 will be made in conjunction with the operation and function illustrated in FIG. 5.</p> <p>Lerner '776 at [0063]: First, at step 501 the user terminal client web browser 412 submits an HTTP request with a cookie to a web application 401 such that the URL of the desired resource (e.g., application 401) available in the central server domain and the cookie is transmitted to the application 401 . Having received the HTTP request and the cookie, at step 502 the application corresponding to the HTTP request (e.g., application 401) transmits the cookie to the application interface library 402 . The application interface library 402 at step 503 then performs user authentication processing. In particular, the application interface library 402 transmits the cookie received from the client web browser 412 to the cookie access library 405 which is configured to read the cookie data at step 504 . In the event that the transmitted cookie is encrypted, the cookie access library 405 transmits the encrypted cookie to the encryption/ decryption engine 403 which, at step 505 decrypts the encrypted cookie and passes the decrypted cookie back to the</p>
--	---

	<p>cookie access library 405 . The cookie access library 405 then parses the decrypted cookie at step 506 into cookie data and accordingly, determines whether the user ID is matched or not by returning a confirm authentication or a null match command to the application interface library 402 . With a matched user ID, the application interface library 402 returns the result to the web application 401 which then delivers the application content at step 507 for display at the client browser 412 at step 508 . On the other hand, if the authentication process is unsuccessful, a null match command is returned to the web application 401 which then delivers redirect information with a HTTP redirect to the client web browser 412 at step 509 . The client web browser 412 then submits the HTTP redirect request to the login page of the application at step 510 , and the login page URL and the cookie is sent to the login page at the client web browser 412 at step 511 where the user is prompted to input login information to access the desired application.</p> <p>Lerner '776 at [0064]:</p> <p>FIG. 6 illustrates a flowchart for a user event processing procedure to access a desired application in accordance with one embodiment of the present invention. As shown, when a user event occurs such as when a user/client requests access to a particular application or resource in the central server domain, at step 601 the user at the user terminal web browser 412 submits an HTTP request corresponding to the requested application or resource, transmitting the corresponding URL and the cookie file. At step 602 , the corresponding web application 401 receives the HTTP request and the cookie file, and in turn, passes the event name and the received cookie to the application interface library 402 which posts the corresponding event at step 603 . In other words, at step 603 , the requested application or data store from the application interface library 402 places the message signifying the user event onto either a browser cookie or a message queue for eventual delivery at step 607 to another (or subsequent) data store or application. Again, the application, event and the user ID in the form of a message can be optionally encrypted at step 608 by the encryption engine 409 when they are placed onto the message queue before being transmitted from the message queuing middleware 404 of the application or data store resident locally at the central server or at a third party vendor site within the central server domain.</p>
--	---

	<p>Lerner '776 at [0070]:</p> <p>Furthermore, the architecture in accordance with the present invention provides a vehicle for different user registration and authentication components to operate together. This feature is invaluable in integrating public web sites and secure site registration authentication and registration, as well as in integrating any outsourced registration/authentication with in house components. With respect to the web applications, any application that can call a library and be location in the central server domain will be able to use the shared services for its registration and authentication. Indeed, the cookie-based authentication allows a user to log in once and move from application to application without the need for re-authentication. Additionally, user behavior and user provided information can be passed to a central user profile repository using the message passing portion of the architecture. The relevant profile can then be placed in the user's browser cookie at the time of user login process. Thus, each application can read the cookie and use it to shape banners, prompts and other content based on the user behavior and provided information.</p>
--	--

Lerner '776 at Fig. 5:

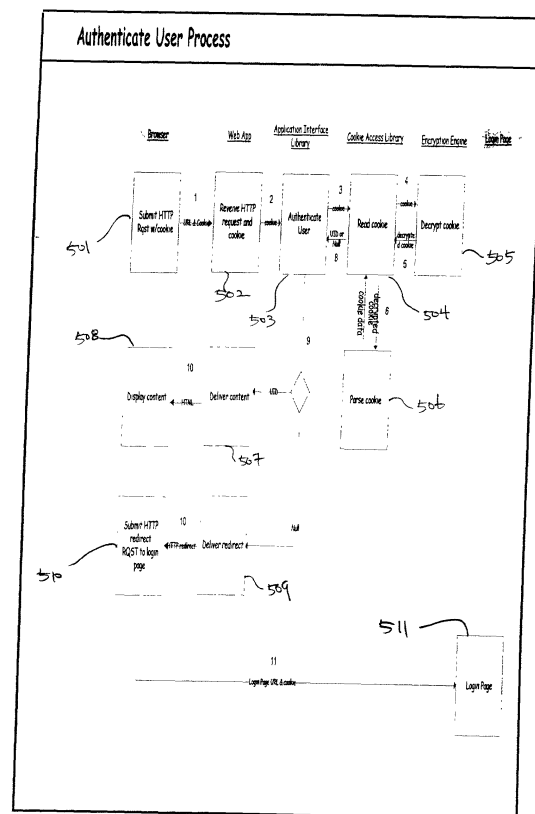


FIGURE 5

Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[9.2] using the user credentials received from the user to authenticate the user.

The Appendix References disclose and/or render obvious: using the user credentials received from the user to authenticate the user.

See, e.g.:

	<p>Wood at p. 15:</p> <p>Once login credentials have been obtained by login component 120, they are supplied (11) to gatekeeper/entry handler component 110 for authentication. In configurations in which both gatekeeper/entry handler component 110 and login component 120 have access to a shared object such as the HttpSession object, login credential passing via the shared object is suitable. In other configurations, an HTTP POST may be employed. In an exemplary configuration, the particular credential passing method is selected as part of the original HTTP redirect (e.g., encoded in the redirect URL) although other configurations need not allow for a selection or may employ other facilities for selection of a credential passing method.</p> <p>Wood at p. 15-16:</p> <p>Login component 120 also passes control of the access request back to gatekeeper/entry handler component 110. In an exemplary configuration, login component 120 issues a new HTTP request (11) specifying the originally requested URL, which has been stored in the HttpSession object. As before, gatekeeper/entry handler component 110 receives the request. Gatekeeper/entry handler component 110 extracts the login credentials from the request or from the HttpSession object and passes (12) the login credentials to authentication component 130 for authentication. Authentication component 130 authenticates the login credential, and if successful, queries (13) identification component 150 to establish a correspondence with a set of entity descriptors that uniquely identifies the requesting entity. In an exemplary configuration, entity descriptor types include: entity id, system id (e.g., name/password), certificate, enigma id, smartcard token, name/address, and phone. One or more of values may uniquely identify an entity and one or more entity descriptor types may support multiple values (e.g., multiple digital certificates, name/password pairs, or phone numbers per identity). Once the requesting entity has been identified (14), session parameters are updated (15) and session management component 160 supplies (16) session credentials. Preferably, session credentials are digitally-signed or otherwise cryptographically secured and returned (17) to gatekeeper/entry handler component 110.</p> <p>Wood at p. 21:</p> <p>If an exception is thrown due to insufficient authorization, e.g., if the signed session credentials do not indicate that the requesting entity is sufficiently authorized to access order management service 312, a login credential gathering process is initiated. Based on the required trust level and on rules that encode the sufficiency of authentication schemes, a login credential is obtained</p>
--	--

	<p>from user 301 and/or browser 302. The obtained login credential is of a type that, if authenticated, is sufficient to meet the trust level requirement for access to order management service 312. Aspects of an exemplary credential gathering process are described in greater detail above. However, as an example, FIG. 3 illustrates the obtaining of a username/password pair. Once login credentials are obtained, they are passed to central security framework 304 (as described above) for authentication by central authentication service 307 so that session credentials can be obtained, the requested access can be authorized, and the order processing initiated. Signed session credentials, typically embodied as a cryptographically secured session token that may be stored as a cookie, are passed back to browser 302 with results of the requested access. In this way, subsequent access requests are identified as part of a session and authorization may be quickly confirmed without unnecessary re-authentication.</p> <p>Lerner '776 at [0010]: While still within schwab.com's server domain, the account holder is nevertheless required to separately log onto the third party vendor's site. Such user authentication or validation is necessary each time the account holder clicks on a hypertext link to be redirected to another information source. Moreover, once the account holder provides the log-in information to the third party vendor, the third party vendor must access back to the central server at schwab.com to verify the account holder's log-in information or to retrieve any other information related to the account holder that may be relevant to the account holder vis a vis the third party vendor. For example, in the case where the account holder is returning to a previous online interactive education course, once the account holder logs onto the third party vendor site, the server at the third party vendor accesses the central server at schwab.com to confirm that the account holder is indeed a Schwab account holder, and furthermore, to determine where in the interactive course the account holder left off during the last session.</p> <p>Lerner '776 at [0023]: FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention.</p> <p>Lerner '776 at [0062]: FIG. 5 illustrates a flowchart for a user authentication process in accordance with one embodiment of the present invention. As shown, the operation and function of the components</p>
--	--

of the architecture for providing the integrated, distributed shared services system illustrated in FIG. 4 is further discussed. Accordingly, reference to the components shown in FIG. 4 will be made in conjunction with the operation and function illustrated in FIG. 5.

Lerner '776 at [0063]:

First, at step 501 the user terminal client web browser 412 submits an HTTP request with a cookie to a web application 401 such that the URL of the desired resource (e.g., application 401) available in the central server domain and the cookie is transmitted to the application 401 . Having received the HTTP request and the cookie, at step 502 the application corresponding to the HTTP request (e.g., application 401) transmits the cookie to the application interface library 402 . The application interface library 402 at step 503 then performs user authentication processing. In particular, the application interface library 402 transmits the cookie received from the client web browser 412 to the cookie access library 405 which is configured to read the cookie data at step 504 . In the event that the transmitted cookie is encrypted, the cookie access library 405 transmits the encrypted cookie to the encryption/ decryption engine 403 which, at step 505 decrypts the encrypted cookie and passes the decrypted cookie back to the cookie access library 405 . The cookie access library 405 then parses the decrypted cookie at step 506 into cookie data and accordingly, determines whether the user ID is matched or not by returning a confirm authentication or a null match command to the application interface library 402 . With a matched user ID, the application interface library 402 returns the result to the web application 401 which then delivers the application content at step 507 for display at the client browser 412 at step 508 . On the other hand, if the authentication process is unsuccessful, a null match command is returned to the web application 401 which then delivers redirect information with a HTTP redirect to the client web browser 412 at step 509 . The client web browser 412 then submits the HTTP redirect request to the login page of the application at step 510 , and the login page URL and the cookie is sent to the login page at the client web browser 412 at step 511 where the user is prompted to input login information to access the desired application.

Lerner '776 at [0064]:

FIG. 6 illustrates a flowchart for a user event processing procedure to access a desired application in accordance with one embodiment of the present invention. As shown, when a user event occurs such as when a user/client requests access to a particular application or

	<p>resource in the central server domain, at step 601 the user at the user terminal web browser 412 submits an HTTP request corresponding to the requested application or resource, transmitting the corresponding URL and the cookie file. At step 602 , the corresponding web application 401 receives the HTTP request and the cookie file, and in turn, passes the event name and the received cookie to the application interface library 402 which posts the corresponding event at step 603 . In other words, at step 603 , the requested application or data store from the application interface library 402 places the message signifying the user event onto either a browser cookie or a message queue for eventual delivery at step 607 to another (or subsequent) data store or application. Again, the application, event and the user ID in the form of a message can be optionally encrypted at step 608 by the encryption engine 409 when they are placed onto the message queue before being transmitted from the message queuing middleware 404 of the application or data store resident locally at the central server or at a third party vendor site within the central server domain.</p> <p>Lerner '776 at [0070]: Furthermore, the architecture in accordance with the present invention provides a vehicle for different user registration and authentication components to operate together. This feature is invaluable in integrating public web sites and secure site registration authentication and registration, as well as in integrating any outsourced registration/authentication with in house components. With respect to the web applications, any application that can call a library and be location in the central server domain will be able to use the shared services for its registration and authentication. Indeed, the cookie-based authentication allows a user to log in once and move from application to application without the need for re-authentication. Additionally, user behavior and user provided information can be passed to a central user profile repository using the message passing portion of the architecture. The relevant profile can then be placed in the user's browser cookie at the time of user login process. Thus, each application can read the cookie and use it to shape banners, prompts and other content based on the user behavior and provided information.</p>
--	---

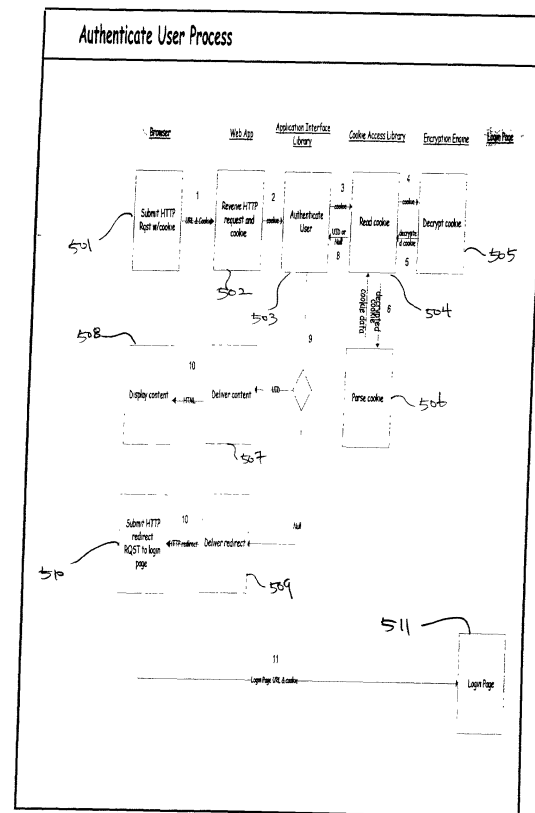


FIGURE 5

Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[10] A computer program product comprising at least one non-transitory computer-readable storage medium storing computer instructions translatable by a

The Appendix References disclose and/or render obvious: A computer program product comprising at least one non-transitory computer-readable storage medium storing computer instructions translatable by a processor of a control device to perform.

See, e.g.:

<p>processor of a control device to perform:</p>	<p>Wood at p. 26-27: 28. A method as in claim 18, embodied as one or more computer program products including functionally descriptive information for directing a processor to perform the login credential obtaining and principal authenticating, the cryptographically secured session credential issuing, and the selectively allowing access based on sufficiency of the first authorization by the cryptographically secured session credential, the one or more computer program products encoded by or transmitted in at least one computer readable medium selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.</p> <p>Wood at p. 27: 32. An access control facility as in claim 29, embodied as a computer program product encoded by or transmitted in at least one computer readable medium selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[10.1] receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user;</p>	<p>The Appendix References disclose and/or render obvious: receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user.</p> <p><i>See</i> [1.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[10.2] processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage;</p>	<p>The Appendix References disclose and/or render obvious: processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage.</p> <p><i>See</i> [1.2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>

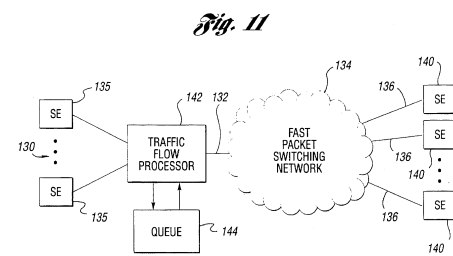
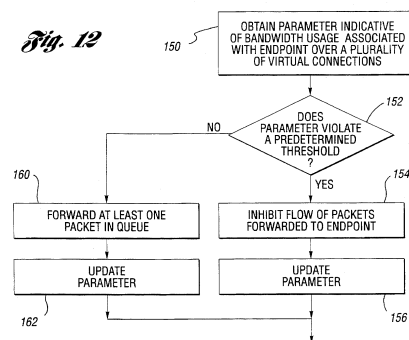
<p>[10.3] at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules; and</p>	<p>The Appendix References disclose and/or render obvious: at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules.</p> <p><i>See</i> [1.3].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[10.4] at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.</p>	<p>The Appendix References disclose and/or render obvious: at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.</p> <p><i>See</i> [1.4].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[11] The computer program product of claim 10, wherein the plurality of stages further comprises an interface specific stage and wherein the computer instructions are further translatable by the processor to perform: at the interface specific stage, applying interface specific rules to the packet based on a type of a network interface over which the packet is received.</p>	<p>The Appendix References disclose and/or render obvious: The computer program product of claim 10, wherein the plurality of stages further comprises an interface specific stage and wherein the computer instructions are further translatable by the processor to perform: at the interface specific stage, applying interface specific rules to the packet based on a type of a network interface over which the packet is received.</p> <p><i>See</i> [2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>

<p>[12] The computer program product of claim 10, wherein the plurality of stages further comprises a global stage and wherein the computer instructions are further translatable by the processor to perform: at the global stage, applying global firewall rules to the packet.</p>	<p>The Appendix References disclose and/or render obvious: The computer program product of claim 10, wherein the plurality of stages further comprises a global stage and wherein the computer instructions are further translatable by the processor to perform: at the global stage, applying global firewall rules to the packet.</p> <p><i>See</i> [3].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[13] The computer program product of claim 10, wherein the computer instructions are further translatable by the processor to implement a control program on the control device, wherein the control program comprises a provisioning module, wherein the provisioning module comprises a traffic conditioning module for user specific allocation of bandwidth.</p>	<p>The Appendix References disclose and/or render obvious: The computer program product of claim 10, wherein the computer instructions are further translatable by the processor to implement a control program on the control device, wherein the control program comprises a provisioning module, wherein the provisioning module comprises a traffic conditioning module for user specific allocation of bandwidth.</p> <p><i>See, e.g.:</i></p> <p>Teraslinna '492 at 1:58-67: Presently, a traffic contract for each virtual connection is negotiated, between a subscriber and a provisioning agent who represents the network, at the time of provisioning or subscription. The terms of the contract include a constraint on the amount of bandwidth which can be used by the subscriber on the virtual connection. The measure of the amount of bandwidth used by a virtual connection can be based upon a measure of the mean bit rate, the peak bit rate, and/or the duration at which the peak rate is sustained, to name a few commonly-used parameters.</p> <p>Teraslinna '492 at 5:1-10: Referring back to FIG. 1, the method further includes a step of enforcing a bandwidth constraint, as indicated by block 12, based upon the source endpoint identified by the address field, wherein the bandwidth constraint is independent of the destination endpoint identified by the address field. In a preferred embodiment wherein the address field contains a source label which identifies the source endpoint and a destination label which identifies the destination endpoint, the bandwidth constraint is enforced based upon the source label and is independent of the destination label.</p>

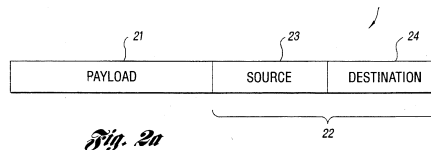
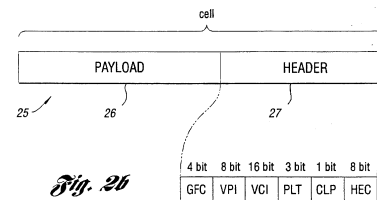
	<p>Teraslinna '492 at 5:61-6:2: The bandwidth constraint for the source endpoint is enforced by measuring the bandwidth usage associated with the source endpoint over all of the plurality of virtual connections. If the packet causes the violation of a predetermined bandwidth constraint, then the packet is penalized as described earlier. In this context, the plurality of virtual connections which emanate from the source endpoint can be considered as a bundle for the purpose of assigning bandwidth thereto and enforcing bandwidth constraints therefor.</p> <p>Teraslinna '492 at 6:24-33: A packet protocol processor 37 performs the packet protocol processing. The packet protocol processor 37 provides means for enforcing a bandwidth constraint based upon an address field contained within the received packet. Consequently, the line unit 34 enforces a bandwidth constraint for packets communicated over a bundle of provisioned virtual connections emanating from a source endpoint identified by the address field. As described earlier, a received packet which violates the bandwidth constraint is penalized in a predetermined manner.</p> <p>Teraslinna '492 at 15:40-49: The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint 132 over at least two of the virtual connections. The at least one parameter indicative of bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.</p> <p>Teraslinna '492 at 15:50-59: The at least one parameter parallels at least one of the bandwidth parameters monitored and enforced by the network 134 in accordance with a predetermined traffic contract. Preferably, the traffic flow processor 142 obtains each of the parameters which are monitored by the network 134 in accordance with the predetermined traffic contract. The at least one parameter may be indicative of an average bit rate or a peak bit rate, for example, associated with the bundle. The at least one parameter may include a leaky bucket parameter as described earlier.</p>
--	---

	<p>Teraslinna '492 at 15:60-67: The traffic flow processor 142 further provides means for inhibiting the flow of packets forwarded to the endpoint 132 when the at least one parameter violates the traffic contract. Typically, this occurs when one of the at least one parameter violates a predetermined threshold. The predetermined threshold may be violated if the parameter exceeds the threshold, or alternatively, if the parameter drops below the threshold.</p> <p>Teraslinna '492 at 16:1-8: The system further includes a queue 144 in communication with the traffic flow processor 142. The queue 144 provides means for storing at least one packet when the traffic contract is violated, i.e. when the parameter violates the predetermined threshold. Preferably, the queue 144 queues all incoming packets from the customer premises equipment 130 for traffic shaping by the traffic flow processor 142.</p>
--	---

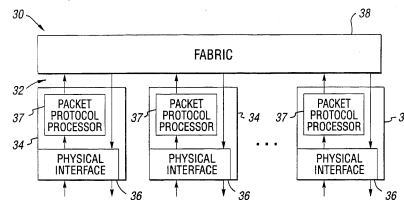
Teraslinna '492 at Fig. 11-12:



Teraslinna '492 at Fig. 2a-2b:



Teraslinna '492 at Fig. 3:



Kadambi '044 at 17:36-51:

Egress manager 76 is also capable of controlling data flow. Each egress manager 76 is provided with the capability to keep track of packet identification information in a packet pointer budget register; as a new pointer is received by egress manager 76, the associated packet pointer budget register is incremented. As egress manager 76 sends out a data packet 112, the packet pointer budget register is decremented. When a storage limit assigned to the register is reached, corresponding to a full packet identification pool, a notification message is sent to all ingress ports of the SOC 10, indicating that the destination egress port controlled by that egress manager 76 is unavailable. When the packet pointer budget register is decremented

	<p>below the packet pool high watermark value, a notification message is sent that the destination egress port is now available. The notification messages are sent by CBM 71 on the S channel 83.</p> <p>Kadambi '044 at 17:52-64: As noted previously, flow control may be provided by CBM 71, and also by ingress submodule 14 of either an EPIC 20, GPIC 30, or by IPIC 90. Ingress submodule 14 monitors cell transmission into port 24. When a data packet 112 is received at a port 24, the ingress submodule 14 increments a received budget register by the cell count of the incoming data packet. When a data packet 112 is sent, the corresponding ingress 14 decrements the received budget register by the cell count of the outgoing data packet 112. The budget register 72 is decremented by ingress 14 in response to a decrement cell count message initiated by CBM 71, when a data packet 112 is successfully transmitted from CBP 50.</p> <p>Kadambi '044 at 37:64-38:33: In conjunction with the filtering functions, the configuration of SOC 10 enables a traffic conditioning function, which can meter, shape, police, drop, and/or remark data packets as necessary, to ensure that the packets or data traffic entering the differentiated services (diffserv) domains conform to predetermined requirements for the particular implementation. Metering functions generally measure the temporal properties, generally the rate or flow, of the stream of packets selected by a classifier. In the present invention a rate counter field for a codepoint in diffserv-to-COS mapping table is incremented every time a packet comes into the switch with that particular codepoint, thus allowing a rate of traffic to be determined. A shaping function serves to delay some or all of the packets in a traffic stream in order to bring the traffic stream into compliance with a predetermined traffic profile. The present invention implements a shaping functionality for each COS queue, which is handled by the COS manager on each individual egress. The dropping function of the metering logic is responsible for discarding some or all of the packets in a data stream in order to bring the data stream into compliance with a predetermined traffic profile. Put simply, if the aforementioned rate counter for a specific code point value exceeds the rate counter threshold set in the diffserv-to-COS table, then the option is provided, using the new codepoint actions bits, to drop the packet from the data stream. The re-marking function allows the codepoint of a packet to be reset depending on the characteristics of the traffic profile. The re-marker may be configured to re-</p>
--	---

	<p>mark all packets to a single codepoint, or it may be configured to mark a packet to one of a set of codepoints. Specifically, if the aforementioned rate counter for a codepoint exceeds the rate counter threshold in the diffserv-toCOS table, then the option is provided, using the new codepoint action bits, to remark the codepoint and select a new COS queue for the packet, in addition to changing the 802.1p priority of the packet, both of which will have a direct impact upon flow threshold of the packet.</p> <p>Kadambi '044 at 38:34-61:</p> <p>Although the present invention provides multiple packet flow control alternatives, metering is generally provided by the counter field of rules table 22. Using the counter and the COS queues, packets in a particular traffic stream can be delayed as necessary in order to bring the traffic stream into compliance with the desired traffic profile. This can be controlled through the use of COS manager 133 and transaction fifo 132, as illustrated in FIG. 13. Packet pointers that depend upon the differentiated services code point (DSCP) are placed in one of the COS queues in transaction fifo 132, and scheduler 134 picks up the next packet depending upon the priority determination of COS manager 133. Queue scheduling algorithms can be programmed into COS manager 133 as appropriate for a particular application. Strict priority based scheduling can be implemented, wherein packets in the high priority COS queue are taken up first for transmission. However, this can result in starvation of low priority COS queues. An option for resolving this difficulty, therefore, is implementing a weighted priority based scheduling scheme, wherein a minimum bandwidth is provided to all COS queues, so that no queue gets starved as a result of priority allocation. Bandwidth is a programmable parameter in COS manager 133, and can be programmed based upon the switch application. Realtime applications can be implemented through a Maximum Allowable Latency Parameter, which enables COS manager 133 to schedule packet transmission such that packets on a particular COS queue are not delayed for more than a maximum allowable latency time.</p> <p>Kadambi '044 at 42:23-62:</p> <p>In another embodiment, the packet flow control logic is folded into the filtering logic, and therefore, the packet flow control logic operates in conjunction with the filtering. In this embodiment, the filtering/flow logic operates in three stages: first, the logic takes actions that are independent of the packet profile, that is the actions do not depend on the classification of the packet as in-profile or out-profile; second, the filtering/flow logic picks up the meterid,</p>
--	--

which is a 6 bit number associated with the packet that is stored in the rules table, and takes any appropriate in-profile actions that are set; and third, the filtering/flow logic takes any appropriate out-profile actions. Beginning with the profile independent actions, upon application of each individual filter mask, which is generally undertaken in ascending numerical order, a determination is made by the filtering/flow logic as to whether there is a matching rule, as shown in FIG. 32. This determination essentially determines whether or not there is a full match, as previously defined, which is shown in FIG. 32 as step 32-1. If it is determined that there is a full match for the particular mask at step 32-1, then the first action bit is checked at step 32-2. If a full match is not determined at step 32-1, then the logic determines whether or not there is a partial match for the mask at step 32-3. If a partial match is found at step 32-3, then the logic continues through the partial match method, which is illustrated in FIG. 33 and will be further discussed below. Returning to step 32-2, if it is determined that bit 1 of the action bits is set, then the class of service is selected from this rule entry at step 32-4. If bit 1 of the action bits is not set, then the logic continues to step 32-5 and checks to see if bit 0 of the action fields is set. If bit 0 is set, then the class of service for this entry is obtained from the rule entry, the packet is modified for priority tagged field, and the regenerate CRC bit is set at step 32-5. If bit 0 is not set, then the logic continues to step 32-6, which generally represents the beginning of the flow control logic, and the end of the profile-independent actions, as action bit 0 and action bit 1 are independent actions from flow control. Put simply, the profile independent actions are taken at steps 32-1 through 32-5, and beginning at step 32-6, the profile dependent actions begin.

Kadambi '044 at 57:50-67:

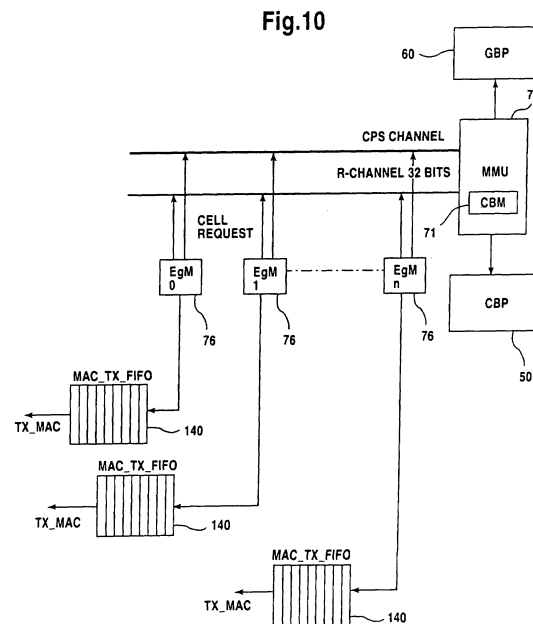
The COS manager 133 can also be programmed using a strict priority based scheduling method, or a weighted priority based scheduling method of selecting the next packet pointer in transaction FIFO 132. Utilizing a strict priority based scheduling method, each of the eight COS priority queues are provided with a priority with respect to each other COS queue. Any packets residing in the highest priority COS queue are extracted from transaction FIFO 132 for transmission. On the other hand, utilizing a weighted priority based scheduling scheme, each COS priority queue is provided with a programmable bandwidth. After assigning the queue priority of each COS queue, each COS priority queue is given a minimum and a maximum bandwidth. The minimum and maximum bandwidth values are user programmable. Once the higher priority queues achieve their minimum bandwidth value, COS manager 133 allocates

any remaining bandwidth based upon any occurrence of exceeding the maximum bandwidth for any one priority queue.

Kadambi '044 at 58:49-57:

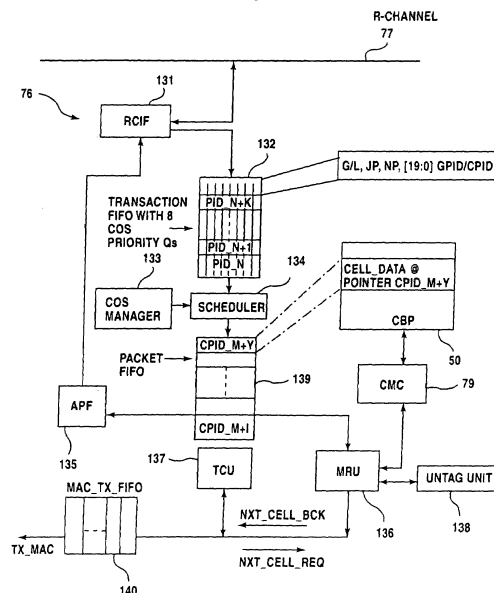
Egress manager 76, through MRU 136, interfaces with transmission FIFO 140, which is a transmission FIFO for an appropriate media access controller (MAC); media access controllers are known in the Ethernet art. MRU 136 prefetches the data packet 112 from the appropriate memory, and sends the packet to transmission FIFO 140, flagging the beginning and the ending of the packet. If necessary, transmission FIFO 140 will pad the packet so that the packet is 64 bytes in length.

Kadambi '044 at Fig. 10:



Kadambi '044 at Fig. 13:

Fig.13



Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[14] The computer program product of claim 10, wherein the computer instructions are further translatable by the processor to establish the user specific traffic control rules and the user specific firewall rules for the user based on a user profile that includes parameters that determine an

The Appendix References disclose and/or render obvious: The computer program product of claim 10, wherein the computer instructions are further translatable by the processor to establish the user specific traffic control rules and the user specific firewall rules for the user based on a user profile that includes parameters that determine an extent of the user's access to the second network, services available to the user, or a combination thereof.

See [7].

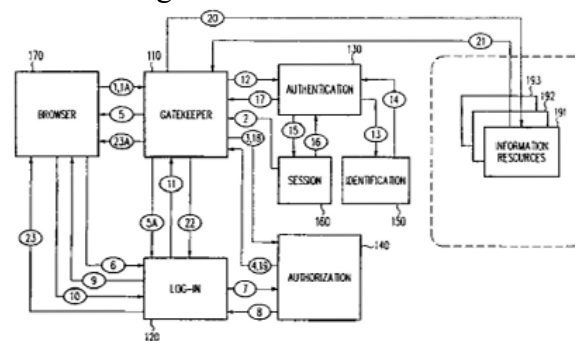
Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

extent of the user's access to the second network, services available to the user, or a combination thereof.	
[15] The computer program product of claim 10, wherein the computer instructions are further translatable by the processor to determine whether the user has authenticated.	<p>The Appendix References disclose and/or render obvious: The computer program product of claim 10, wherein the computer instructions are further translatable by the processor to determine whether the user has authenticated.</p> <p><i>See</i> [8].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[16] The computer program product of claim 15, wherein the user has not authenticated and wherein the computer instructions are further translatable by the processor to perform:	<p>The Appendix References disclose and/or render obvious: The computer program product of claim 15, wherein the user has not authenticated and wherein the computer instructions are further translatable by the processor to perform.</p> <p><i>See</i> [9] (preamble).</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[16.1] redirecting the user to a login web page that requests user credentials; and	<p>The Appendix References disclose and/or render obvious: redirecting the user to a login web page that requests user credentials; and.</p> <p><i>See</i> [9.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[16.2] using the user credentials received from the user to authenticate the user.	<p>The Appendix References disclose and/or render obvious: using the user credentials received from the user to authenticate the user.</p> <p><i>See</i> [9.2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>

<p>[17] A control device for network access control, comprising:</p>	<p>The Appendix References disclose and/or render obvious: A control device for network access control.</p> <p><i>See, e.g.:</i></p> <p>Wood at p. 6: FIG. 1 provides an overview of major interactions between components for an exemplary security architecture in accordance with the present invention. As illustrated in FIG. 1, a client application, e.g., a browser 170 operated by a user, interacts with the security architecture via a gatekeeper and entry handler component 110 and a login component 120. Gatekeeper and entry handler component 110 provides an entry point for external client applications requesting access to enterprise applications and/or resources, including e.g., information resources 191, 192 193, for which access management is provided by the security architecture. Using facilities provided by a session management component 160, an authorization component 140, an authentication component 130, an identification component 150, and login component 120, the gatekeeper/entry handler component 110 allows, redirects or refuses access requests in accordance with a security policy.</p> <p>Wood at p. 6-7: Login component 120, operating in conjunction with gatekeeper/entry handler component 110 and other components of the security architecture, provides a single sign-on interface for access to enterprise applications and/or resources. In an exemplary embodiment, security requirements are expressed in terms of trust levels and login component 120 obtains login credentials for an entity requesting access to one of the enterprise applications and/or resources. The login credentials obtained are selected from a set of credential types that, if authenticated, are sufficient to achieve the trust level requirement of an application or information resource to be accessed. Without limitation, typical login credential types and associated authentication mechanisms include those based on passwords, certificates, biometric techniques, smart cards, etc. Other credential types and associated authentication mechanisms are described elsewhere herein.</p> <p>Wood at p. 11: Entry handler functionality (e.g., in gatekeeper/entry handler component 110) ascertains much of the requesting client's environment information. For example, for dial-up connections, environment information such as line speed and low-level line encryption are ascertained.</p>
--	--

Additional information, such as source number (e.g., from caller id information or based on a callback configuration), signaling type (e.g., POTS or digital ISDN), etc., may also be obtained. For network connections, similar environment information (e.g., source network and/or node, Virtual Private Network (VPN) low-level encryption, etc.) may be obtained from incoming requests themselves or based on a particular entry point (e.g., a particular router or port). More generally, gatekeeper/entry handler component 110 obtains and/or maintains information such as connect location (if ascertainable), temporal information such as request time/date, session start time/date, etc. (preferably in both the client entity's frame of reference and the security architecture or requested information resource's frame of reference, if location is ascertainable), and client type and/or capabilities (e.g., browser type and Java Development Kit (JDK) level). In some configurations, information on line speed, origination point (e.g., inside or outside of a corporate network), browser type, encryption capability, number of hops, latency, system type, etc. may be gathered. Such information is used in some configurations for mapping particular authentication mechanisms to trust levels and for authorization decisions. Environment information is generally packaged into a data structure that is associated with a client session. Other components of the security architecture may add additional client environment information, such as authentication strength or current trust level.

Wood at Fig. 1:



Teraslinna '492 at 15:40-49:

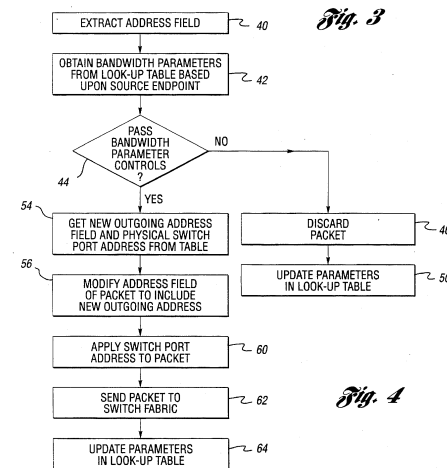
The system includes a traffic flow processor 142 operatively associated with the customer premises equipment 130 and the network 134. The traffic flow processor 142 provides means for obtaining at least one parameter indicative of bandwidth usage associated with the endpoint

132 over at least two of the virtual connections. The at least one parameter indicative of bandwidth usage provides a measure of the bandwidth associated with a bundle of provisioned virtual connections, the bundle consisting of the at least two of the virtual connections.

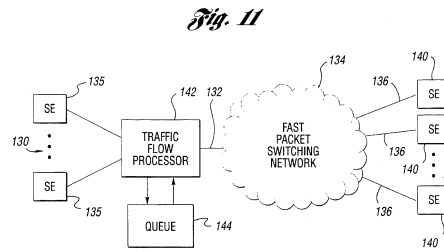
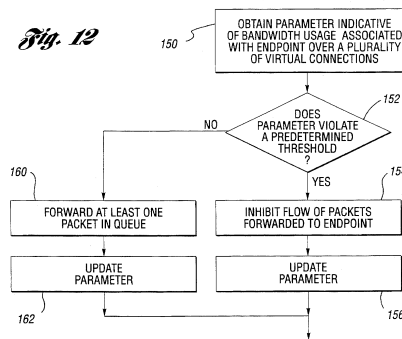
Teraslinna '492 at 15:60-67:

The traffic flow processor 142 further provides means for inhibiting the flow of packets forwarded to the endpoint 132 when the at least one parameter violates the traffic contract. Typically, this occurs when one of the at least one parameter violates a predetermined threshold. The predetermined threshold may be violated if the parameter exceeds the threshold, or alternatively, if the parameter drops below the threshold.

Teraslinna '492 at Fig. 4:



Teraslinna '492 at Fig. 11-12:



Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[17.1] a processor;

The Appendix References disclose and/or render obvious: a processor.

See, e.g.:

Wood at p. 26-27:

28. A method as in claim 18, embodied as one or more computer program products including functionally descriptive information for directing a processor to perform the login credential obtaining and principal authenticating, the cryptographically secured session credential issuing, and the selectively allowing access based on sufficiency of the first authorization by the cryptographically secured session credential, the one or more computer program products

	<p>encoded by or transmitted in at least one computer readable medium selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[17.2] at least one computer-readable storage medium storing computer instructions translatable by the processor to perform:	<p>The Appendix References disclose and/or render obvious: at least one computer-readable storage medium storing computer instructions translatable by the processor to perform.</p> <p><i>See</i> [10] (preamble).</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[17.3] receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user;	<p>The Appendix References disclose and/or render obvious: receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user.</p> <p><i>See</i> [1.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[17.4] processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage;	<p>The Appendix References disclose and/or render obvious: processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage.</p> <p><i>See</i> [1.2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[17.5] at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user	<p>The Appendix References disclose and/or render obvious: at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules.</p> <p><i>See</i> [1.3].</p>

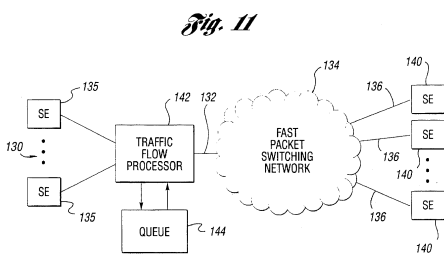
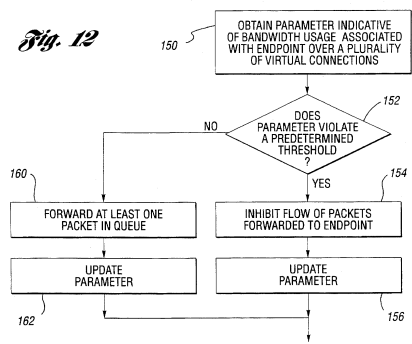
specific traffic control rules and user specific firewall rules; and	Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.
[17.6] at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.	<p>The Appendix References disclose and/or render obvious: at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.</p> <p><i>See</i> [1.4].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[18] The control device of claim 17, wherein the plurality of stages further comprises an interface specific stage and wherein the computer instructions are further translatable by the processor to perform: at the interface specific stage, applying interface specific rules to the packet based on a type of a network interface over which the packet is received.	<p>The Appendix References disclose and/or render obvious: The control device of claim 17, wherein the plurality of stages further comprises an interface specific stage and wherein the computer instructions are further translatable by the processor to perform: at the interface specific stage, applying interface specific rules to the packet based on a type of a network interface over which the packet is received.</p> <p><i>See</i> [2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[19] The control device of claim 17, wherein the plurality of stages further comprises a global stage and wherein the computer instructions are further translatable by the processor to perform: at the global stage,	<p>The Appendix References disclose and/or render obvious: The control device of claim 17, wherein the plurality of stages further comprises a global stage and wherein the computer instructions are further translatable by the processor to perform: at the global stage, applying global firewall rules to the packet.</p> <p><i>See</i> [3].</p>

applying global firewall rules to the packet.	Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.
[20] The control device of claim 17, wherein the computer instructions are further translatable by the processor to implement a control program on the control device, wherein the control program comprises a provisioning module, wherein the provisioning module comprises a traffic conditioning module for user specific allocation of bandwidth.	<p>The Appendix References disclose and/or render obvious: The control device of claim 17, wherein the computer instructions are further translatable by the processor to implement a control program on the control device, wherein the control program comprises a provisioning module, wherein the provisioning module comprises a traffic conditioning module for user specific allocation of bandwidth.</p> <p><i>See</i> [13].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[21] The control device of claim 20, wherein the traffic conditioning module further comprises an interface master queue for controlling a flow of network traffic over a particular network interface.	<p>The Appendix References disclose and/or render obvious: The control device of claim 20, wherein the traffic conditioning module further comprises an interface master queue for controlling a flow of network traffic over a particular network interface.</p> <p><i>See, e.g.:</i></p> <p>Teraslinna '492 at 7:64-8:9: Using embodiments of the present invention, bandwidth use is monitored and enforced for each of the three source endpoints 82a-82c rather than for each of the 15 virtual connections. In terms of the bundling terminology: the five virtual connections from the endpoint 82a are considered as a first bundle; the five virtual connections from the endpoint 82b are considered as a second bundle; and the five virtual connections from the endpoint 82c are considered as a third bundle. As a result, less bandwidth needs to be allocated in the transmission facility 76 to provide the 15 virtual connections in three bundles, in comparison to having 15 provisioned virtual connections each having bandwidth associated therewith.</p> <p>Teraslinna '492 at 16:1-8: The system further includes a queue 144 in communication with the traffic flow processor 142. The queue 144 provides means for storing at least one packet when the traffic contract is</p>

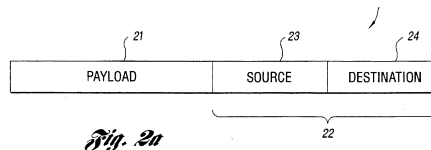
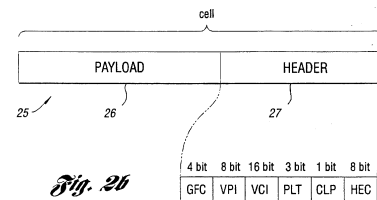
	<p>violated, i.e. when the parameter violates the predetermined threshold. Preferably, the queue 144 queues all incoming packets from the customer premises equipment 130 for traffic shaping by the traffic flow processor 142.</p> <p>Teraslinna '492 at 16:13-21: When the at least one parameter complies with the predetermined traffic contract, i.e., each parameter complies with a corresponding predetermined threshold, the traffic flow processor 142 performs a step of forwarding one of the at least one packet stored in the queue 144 to the endpoint 132. The traffic flow processor 142 further updates each of the at least one parameter when the packet is forwarded to the endpoint 132. Typically, the leaky bucket parameter is updated by performing a step of incrementing the parameter.</p> <p>Teraslinna '492 at 16:31-41: FIG. 12 is a flow chart of an embodiment of a method of controlling a flow of packets being forwarded by a customer premises equipment to an endpoint of a fast packet switching network. The fast packet switching network has a plurality of virtual connections provisioned from the endpoint to a plurality of destination endpoints, wherein the virtual connections are utilized for communicating the packets to the destination endpoints. The packets provided by the customer premises equipment are stored in a queue, and presented to the endpoint in accordance with the following method.</p> <p>Teraslinna '492 at 16:51-58: If the parameter violates a predetermined threshold specified by a predetermined traffic contract, as determined by block 152, a step of inhibiting the flow of packets being forwarded to the endpoint is performed, as indicated by block 154. Consequently, when the flow is inhibited, the number of packets stored in the queue increases as further packets are provided by the customer premises equipment.</p> <p>Teraslinna '492 at 16:64-17:8: If the parameter does not violate the predetermined threshold, i.e., if the parameter complies with the predetermined threshold as determined by block 152, a step of forwarding one of the at least one packet in the queue is performed, as indicated by block 160. A step of updating the parameter when the packet is forwarded to the endpoint from the customer premise equipment is performed, as indicated by block 162. If a leaky bucket parameter is employed,</p>
--	--

	<p>the parameter is incremented in response to the packet being forwarded to the endpoint. Further, the parameter may be decremented in accordance with a step of decrementing the parameter at periodic intervals.</p> <p>Teraslinna '492 at 17:9-18: By repeated executing the above-described series of steps, packets stored within the queue are successively forwarded to the endpoint until it is deemed that forwarding of a subsequent packet is in violation of the predetermined traffic contract. At this point, the length of the queue begins to grow as further packets are provided by the CPE. The bandwidth parameter is repeatedly updated until a time when the parameter no longer violates the traffic contract. At this time, the subsequent packet (along with other packets in compliance with the contract) is forwarded to the endpoint.</p>
--	--

Teraslinna '492 at Fig. 11-12:

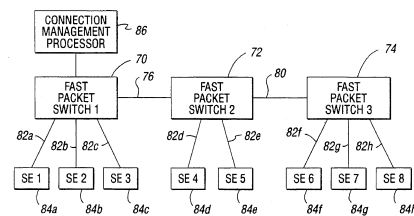


Teraslinna '492 at Fig. 2a-2b:



Teraslinna '492 at Fig. 5:

U.S. Patent Apr. 22, 1997 Sheet 3 of 6 5,623,492



Lerner '776 at [0043]:

Moreover, similar to the embodiment of FIG. 2, the encryption/decryption engine 340 is provided to encrypt and decrypt data and may be omitted for bypassed if the data does not contain sensitive information. The message queuing middleware 350 is configured to package data into messages and assure their delivery, even over unreliable transport media such as the

	<p>internet. In the case where assured delivery is not required, the queuing portion of the operation shown in FIG. 3 may be omitted and only the message packaging function may be used. This approach may be result in improved performance or reduced operational burdens on other servers.</p> <p>Lerner '776 at [0048]: The types of data sharing discussed above should be initiated by the original web application 310 by simply posting the event via the application interface library 330 , or alternatively, via an interface adapter 320 which is configured to monitor the changes to predetermined tables or values in the internal database. The application interface library 330 formats the message and places it in a queue for transmission over the internet even if communication is temporarily disrupted. Additionally, the message can optionally be encrypted by the encryption/decryption engine 340 before the transmission to preserve the security of the message being transmitted.</p> <p>Lerner '776 at [0050]: Accordingly, in accordance with one embodiment of the present invention, when a user event of interest occurs on a web or other application 310 , the application 310 calls a function in the application interface library 330 to notify the “system” of the occurrence of the event. The application 310 then passes one or more NAME=VALUE pairs as parameters to indicate the type of event that has occurred and the relevant data related to the event. Alternatively, the interface adapter 320 directly detects the occurrence of the event by monitoring the internal database (which can be a system file or a memory location) of the application 310 and calls the application interface library 330 for the appropriate function. The application interface library 330 then calls the message queuing middleware 350 to package the data for transmission to the message broker 390 . If necessary, the message is first encrypted by the encryption/decryption engine 340 . In the case where assured delivery is not necessary, the queuing portion of the message queuing middleware 350 may be disabled for the message; otherwise the message is queued up for delivery as soon as the communication path is open or available. If the transport medium is reliable, local and secure, the message queuing middleware 350 may be bypassed and the application interface library 330 may communicate directly to the message broker 390 or to another application interface library on another application or database.</p>
--	---

	<p>Lerner '776 at [0075]:</p> <p>Furthermore, the application interface library and the cookie based portion of the architecture can be deployed on multiple web servers set up for automatic rollover. Leading message brokers and message oriented middleware products can also be set up with multiple servers or multiple queues designed to failover automatically in the event of planned downtime or unanticipated system failure. Since the underlying data transport is the internet, if a user can access an application from client terminal browser, the user access can also pass data to the application via its cookies. The message queuing middleware is designed to hold onto data in the event of a network failure, and to resume transmission automatically when the connection is restored.</p>
--	---

Lerner '776 at Fig. 3:

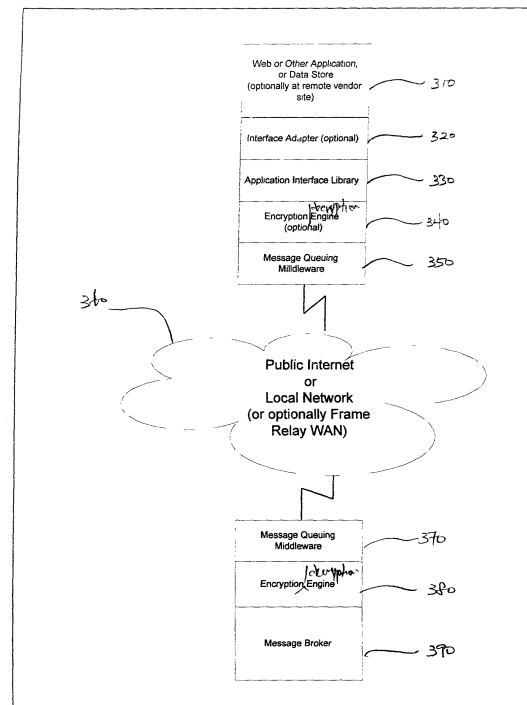


FIGURE 3

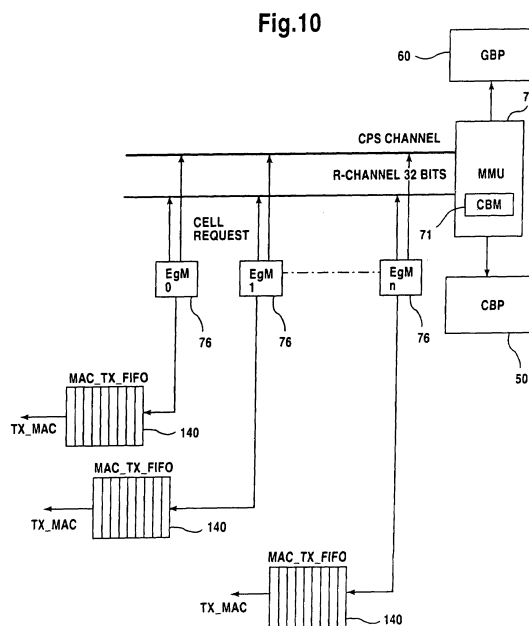
Kadambi '044 at 57:14-49:

Each egress manager 76 includes an R channel interface unit (RCIF) 131, a transaction FIFO 132, a COS manager 133, a scheduler 134, an accelerated packet flush unit (APF) 135, a memory read unit (MRU) 136, a time stamp check unit (TCU) 137, and an untag unit 138. MRU 136 communicates with CBP Memory Controller (CMC) 79, which is connected to CBP 50. Scheduler 134 is connected to a packet FIFO 139. RCIF 131 handles all messages between CBM 71 and egress manager 76. When a packet 112 is received and stored in SOC 10, CBM 71 passes the packet information to RCIF 131 of the associated egress manager 76. The packet

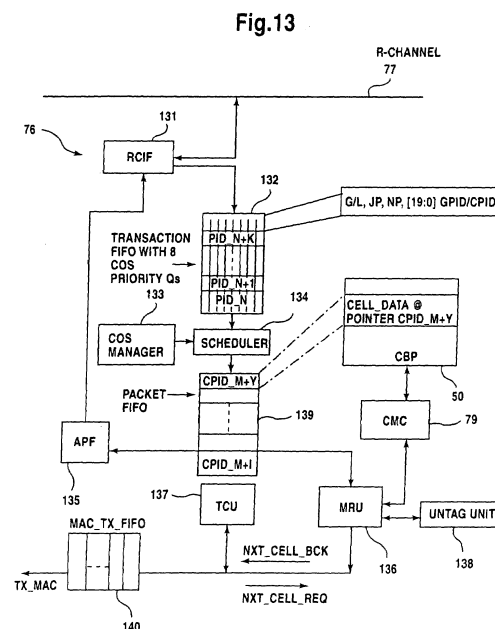
	<p>information will include an indication of whether or not the packet is stored in CBP 50 or GBP 70, the size of the packet, and the PID. RCIF 131 then passes the received packet information to transaction FIFO 132. Transaction FIFO 132 is a fixed depth FIFO with eight COS priority queues, and is arranged as a matrix with a number of rows and columns. Each column of transaction FIFO 132 represents a class of service (COS), and the total number of rows equals the number of transactions allowed for any one class of service. COS manager 133 works in conjunction with scheduler 134 in order to provide policy based quality of service (QOS), based upon Ethernet standards. As data packets arrive in one or more of the COS priority queues of transaction FIFO 132, scheduler 134 directs a selected packet pointer from one of the priority queues to the packet FIFO 139. The selection of the packet pointer is based upon a queue scheduling algorithm, which is programmed by a user through CPU 52, within COS manager 133. An example of a COS issue is video, which requires greater bandwidth than text documents. A data packet 112 of video information may therefore be passed to packet FIFO 139 ahead of a packet associated with a text document. The COS manager 133 would therefore direct scheduler 134 to select the packet pointer associated with the packet of video data.</p> <p>Kadambi '044 at 58:25-33: APF 135 monitors the status of packet FIFO 139. After packet FIFO 139 is full for a specified time period, APF 135 flushes out the packet FIFO. The CBM reclaim unit is provided with the packet pointers stored in packet FIFO 139 by APF 135, and the reclaim unit is instructed by APF 135 to release the packet pointers as part of the free address pool. APF 135 also disables the ingress port 21 associated with the egress manager 76.</p> <p>Kadambi '044 at 58:49-57: Egress manager 76, through MRU 136, interfaces with transmission FIFO 140, which is a transmission FIFO for an appropriate media access controller (MAC); media access controllers are known in the Ethernet art. MRU 136 prefetches the data packet 112 from the appropriate memory, and sends the packet to transmission FIFO 140, flagging the beginning and the ending of the packet. If necessary, transmission FIFO 140 will pad the packet so that the packet is 64 bytes in length.</p> <p>Kadambi '044 at 70:54-71:7: Upon completion of the port switch, additional logic is implemented on the egress side of the switch in order to complete the port switching logic. FIG. 53 illustrates the logic on the egress</p>
--	--

side of the switch, beginning with step 53-1. At step 53-1 a frame is sent out on the egress port, thereby removing a frame from one of the first or second queues. Once a frame is sent out on the egress, then the logic continues to step 53-2, where it is determined if the frame originated from an first queue. If the frame did not originate from an first queue, then the logic continues to step 53-3, where a determination is made as to whether there is another frame queued in a first or second queue. If another frame is waiting for egress in one of the queues, then the logic returns to step 53-1 to send the awaiting frame out on the egress. If no more frames are queued for egress, then the logic continues to step 53-7, where the logic awaits for a frame to arrive in one of the queues. Thereafter, at step 53-8 the logic determines if there are any frames in the first queue. If so, the logic determines if it is ok to send the frame out on the egress, and if not, then the logic returns to the initial state at step 53-1.

Kadambi '044 at Fig. 10:



Kadambi '044 at Fig. 13:



Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[22] The control device of claim 20, wherein the provisioning module further comprises a firewall module for associating the packet with the user specific firewall rules.

The Appendix References disclose and/or render obvious: The control device of claim 20, wherein the provisioning module further comprises a firewall module for associating the packet with the user specific firewall rules.

The Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.

[23] The control device of claim 20, wherein the provisioning module further comprises an

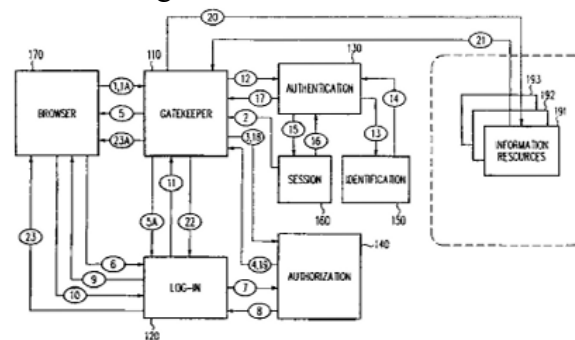
The Appendix References disclose and/or render obvious: The control device of claim 20, wherein the provisioning module further comprises an authentication module for authenticating the user.

<p>authentication module for authenticating the user.</p>	<p><i>See, e.g.:</i></p> <p>Wood at p. 6-7: Login component 120, operating in conjunction with gatekeeper/entry handler component 110 and other components of the security architecture, provides a single sign-on interface for access to enterprise applications and/or resources. In an exemplary embodiment, security requirements are expressed in terms of trust levels and login component 120 obtains login credentials for an entity requesting access to one of the enterprise applications and/or resources. The login credentials obtained are selected from a set of credential types that, if authenticated, are sufficient to achieve the trust level requirement of an application or information resource to be accessed. Without limitation, typical login credential types and associated authentication mechanisms include those based on passwords, certificates, biometric techniques, smart cards, etc. Other credential types and associated authentication mechanisms are described elsewhere herein.</p> <p>Wood at p. 8: As described above, login credentials (e.g., represented in a form such as exemplary login credentials structure 410) are obtained for a client entity. Typically, login credentials encoded in login credentials structure 410 are obtained from a principal via browser client and serve as evidence that the principal (e.g., a human user) is entitled to a particular identity. Accordingly, login credentials structure 410 encodes a userId and a domainId within which the userId should uniquely correspond to a principal. Specific login credentials, e.g., a password, a certificate, results of a biometric process, a response to an Enigma challenge or results of a smart card interrogation, etc. are encoded in login credentials structure 410, as a tagged value. An authenticationScheme is specified and creation time may be encoded to limit replay attacks. In the implementation of FIG. 4, login credentials structure 410 is encrypted using the public key of an authentication service (e.g., of authentication component 130). Because the key is public, any component, even untrusted components may encrypt login credentials for provision to authentication component 130, while only authentication component can decrypt the encrypted login credentials using its private key. In some configurations, secure transfer protocols, e.g., SSL, are employed to secure login credentials between a client entity such as browser 170 and the security architecture while encryption with a public key of an authentication service is performed within the security architecture, e.g., at login component 120. In other configurations, encryption with a public key of an authentication service may be performed at the client entity.</p>
---	---

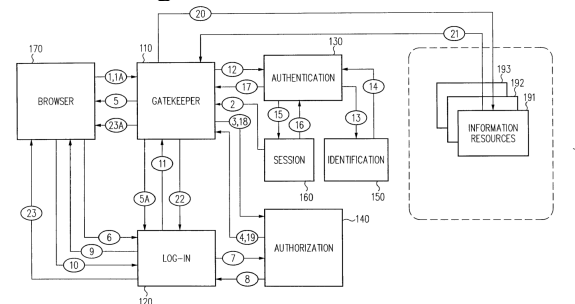
	<p>Wood at p. 15:</p> <p>Once login credentials have been obtained by login component 120, they are supplied (11) to gatekeeper/entry handler component 110 for authentication. In configurations in which both gatekeeper/entry handler component 110 and login component 120 have access to a shared object such as the HttpSession object, login credential passing via the shared object is suitable. In other configurations, an HTTP POST may be employed. In an exemplary configuration, the particular credential passing method is selected as part of the original HTTP redirect (e.g., encoded in the redirect URL) although other configurations need not allow for a selection or may employ other facilities for selection of a credential passing method.</p> <p>Wood at p. 15-16:</p> <p>Login component 120 also passes control of the access request back to gatekeeper/entry handler component 110. In an exemplary configuration, login component 120 issues a new HTTP request (11) specifying the originally requested URL, which has been stored in the HttpSession object. As before, gatekeeper/entry handler component 110 receives the request. Gatekeeper/entry handler component 110 extracts the login credentials from the request or from the HttpSession object and passes (12) the login credentials to authentication component 130 for authentication. Authentication component 130 authenticates the login credential, and if successful, queries (13) identification component 150 to establish a correspondence with a set of entity descriptors that uniquely identifies the requesting entity. In an exemplary configuration, entity descriptor types include: entity id, system id (e.g., name/password), certificate, enigma id, smartcard token, name/address, and phone. One or more of values may uniquely identify an entity and one or more entity descriptor types may support multiple values (e.g., multiple digital certificates, name/password pairs, or phone numbers per identity). Once the requesting entity has been identified (14), session parameters are updated (15) and session management component 160 supplies (16) session credentials. Preferably, session credentials are digitally-signed or otherwise cryptographically secured and returned (17) to gatekeeper/entry handler component 110.</p> <p>Wood at p. 21:</p> <p>If an exception is thrown due to insufficient authorization, e.g., if the signed session credentials do not indicate that the requesting entity is sufficiently authorized to access order management service 312, a login credential gathering process is initiated. Based on the required trust level and on rules that encode the sufficiency of authentication schemes, a login credential is obtained</p>
--	--

from user 301 and/or browser 302. The obtained login credential is of a type that, if authenticated, is sufficient to meet the trust level requirement for access to order management service 312. Aspects of an exemplary credential gathering process are described in greater detail above. However, as an example, FIG. 3 illustrates the obtaining of a username/password pair. Once login credentials are obtained, they are passed to central security framework 304 (as described above) for authentication by central authentication service 307 so that session credentials can be obtained, the requested access can be authorized, and the order processing initiated. Signed session credentials, typically embodied as a cryptographically secured session token that may be stored as a cookie, are passed back to browser 302 with results of the requested access. In this way, subsequent access requests are identified as part of a session and authorization may be quickly confirmed without unnecessary re-authentication.

Wood at Fig. 1:



Wood at Fig. 2:



	Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.
[24] The control device of claim 17, wherein the computer instructions are further translatable by the processor to establish the user specific traffic control rules and the user specific firewall rules for the user based on a user profile that includes parameters that determine an extent of the user's access to the second network, services available to the user, or a combination thereof.	<p>The Appendix References disclose and/or render obvious: The control device of claim 17, wherein the computer instructions are further translatable by the processor to establish the user specific traffic control rules and the user specific firewall rules for the user based on a user profile that includes parameters that determine an extent of the user's access to the second network, services available to the user, or a combination thereof.</p> <p><i>See</i> [7].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[25] The control device of claim 17, wherein the computer instructions are further translatable by the processor to determine whether the user has authenticated.	<p>The Appendix References disclose and/or render obvious: The control device of claim 17, wherein the computer instructions are further translatable by the processor to determine whether the user has authenticated.</p> <p><i>See</i> [8].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[26] The control device of claim 25, wherein the user has not authenticated and wherein the computer instructions are further translatable by the processor to perform:	<p>The Appendix References disclose and/or render obvious: The control device of claim 25, wherein the user has not authenticated and wherein the computer instructions are further translatable by the processor to perform.</p> <p><i>See</i> [9] (preamble).</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>

<p>[26.1] redirecting the user to a login web page that requests user credentials; and</p>	<p>The Appendix References disclose and/or render obvious: redirecting the user to a login web page that requests user credentials.</p> <p><i>See</i> [9.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[26.2] using the user credentials received from the user to authenticate the user.</p>	<p>The Appendix References disclose and/or render obvious: using the user credentials received from the user to authenticate the user.</p> <p><i>See</i> [9.2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27] A control device for network access control, comprising:</p>	<p>The Appendix References disclose and/or render obvious: A control device for network access control.</p> <p><i>See</i> [17] (preamble).</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27.1] a processor;</p>	<p>The Appendix References disclose and/or render obvious: a processor.</p> <p><i>See</i> [17.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27.2] at least one computer-readable storage medium storing computer instructions translatable by the processor to perform:</p>	<p>The Appendix References disclose and/or render obvious: at least one computer-readable storage medium storing computer instructions translatable by the processor to perform.</p> <p><i>See</i> [10] (preamble).</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>

<p>[27.3] receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user;</p>	<p>The Appendix References disclose and/or render obvious: receiving a packet originating from a user device in a first network, wherein the first network is connected to a second network via the control device, and wherein the user device is associated with a user.</p> <p><i>See</i> [1.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27.4] determining whether the user has authenticated;</p>	<p>The Appendix References disclose and/or render obvious: determining whether the user has authenticated.</p> <p><i>See</i> [8].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27.5] if the user has not authenticated:</p>	<p>The Appendix References disclose and/or render obvious: if the user has not authenticated.</p> <p><i>See</i> [9] (preamble).</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27.5.1] redirecting the user to a login web page that requests user credentials; and</p>	<p>The Appendix References disclose and/or render obvious: redirecting the user to a login web page that requests user credentials.</p> <p><i>See</i> [9.1].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
<p>[27.5.2] using the user credentials received from the user to authenticate the user;</p>	<p>The Appendix References disclose and/or render obvious: using the user credentials received from the user to authenticate the user.</p> <p><i>See</i> [9.2].</p>

	Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.
[27.6] processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage;	<p>The Appendix References disclose and/or render obvious: processing the packet according to a plurality of stages, including a client discrimination stage and a user specific rule stage.</p> <p><i>See</i> [1.2].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[27.7] at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules; and	<p>The Appendix References disclose and/or render obvious: at the client discrimination stage, extracting information associated with the user device from a header of the packet and associating the packet with user specific traffic control rules and user specific firewall rules.</p> <p><i>See</i> [1.3].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>
[27.8] at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.	<p>The Appendix References disclose and/or render obvious: at the user specific rule stage, accessing the user specific traffic control rules and user specific firewall rules based on the extracted information associated with the user device and applying the user specific traffic control rules and the user specific firewall rules to the packet as governed by at least one user specific class of service rule associated with the user on the user device in the first network.</p> <p><i>See</i> [1.4].</p> <p>Furthermore, the Appendix References render this limitation obvious in combination with the additional prior art cited in Fortinet's Invalidity Contentions.</p>