US009270701B1

(12) **United States Patent**
Lamb et al.

(10) **Patent No.: US 9,270,701 B1**
(45) **Date of Patent: Feb. 23, 2016**

(54) **SYSTEM AND METHODS FOR USAGE MANAGEMENT IN MULTI-LEVEL SECURITY NETWORKS**

(71) Applicant: **STC.UNM**, Albuquerque, NM (US)

(72) Inventors: **Christopher C. Lamb**, Albuquerque, NM (US); **Gregory L. Heileman**, Albuquerque, NM (US)

(73) Assignee: **STC.UNM**, Albuquerque, NM (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 231 days.

(21) Appl. No.: **13/871,250**

(22) Filed: **Apr. 26, 2013**

Related U.S. Application Data

(60) Provisional application No. 61/639,162, filed on Apr. 27, 2012.

(51) **Int. Cl.**
G06F 17/00 (2006.01)
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/20*** (2013.01)

(58) **Field of Classification Search**
USPC 726/1
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,865,611	B1 *	3/2005	Bragg	709/238
8,621,573	B2 *	12/2013	Bagepalli et al.	726/4
8,824,370	B2 *	9/2014	McNamee et al.	370/328
2003/0076955	A1 *	4/2003	Alve et al.	380/201
2007/0101018	A1 *	5/2007	Shirazipour et al.	709/238
2007/0110009	A1 *	5/2007	Bachmann et al.	370/338
2009/0086959	A1 *	4/2009	Irwin et al.	379/266.01
2009/0216906	A1 *	8/2009	Weniger et al.	709/246

OTHER PUBLICATIONS

Lamb et al., Overlay Architectures enabling Cloud Computing for Multi-Level Security, 2012 IEEE Eighth World Congress on Services Environments.*

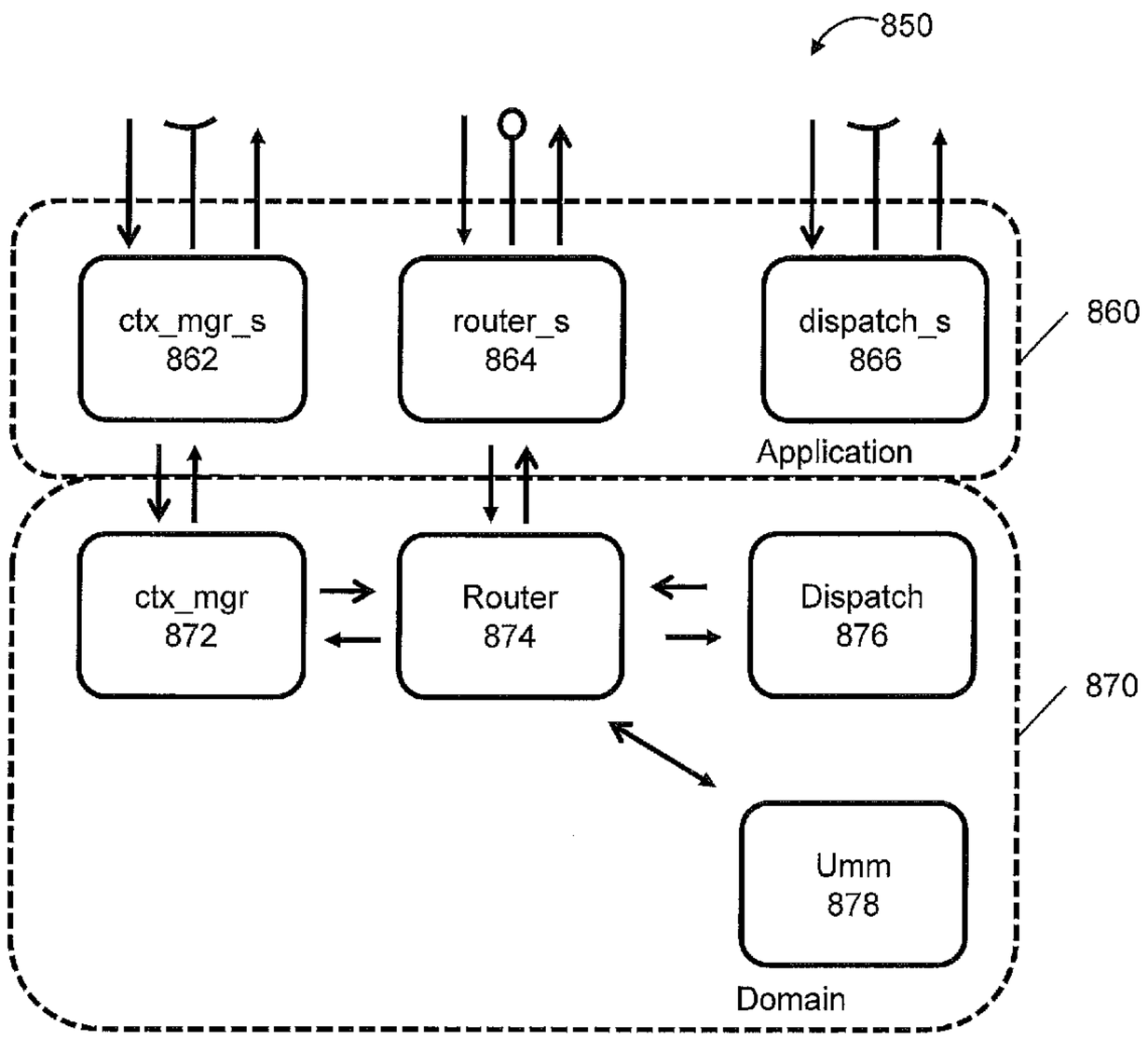
* cited by examiner

Primary Examiner — Amare F Tabor
(74) *Attorney, Agent, or Firm* — Valauskas Corder LLC

(57) **ABSTRACT**

A system and methods for transferring information between two or more incompatible security domains or levels of classification by embedding policy-centric content management components into an information-centric network. Specifically, overlay architectures enable cloud computing for multi-level security environments.

7 Claims, 15 Drawing Sheets



Name	*Description*
ϕ	The initial level of this taxonomy, ϕ classified systems have a single guard without policy-based control
α	α classified systems have a single guard by have begun to integrate policy-based control
β	Systems that have begun to integrate policy-based control with router elements are in the β category
γ	Systems that have integrated policy-based control with routing and computational elements

FIG. 1

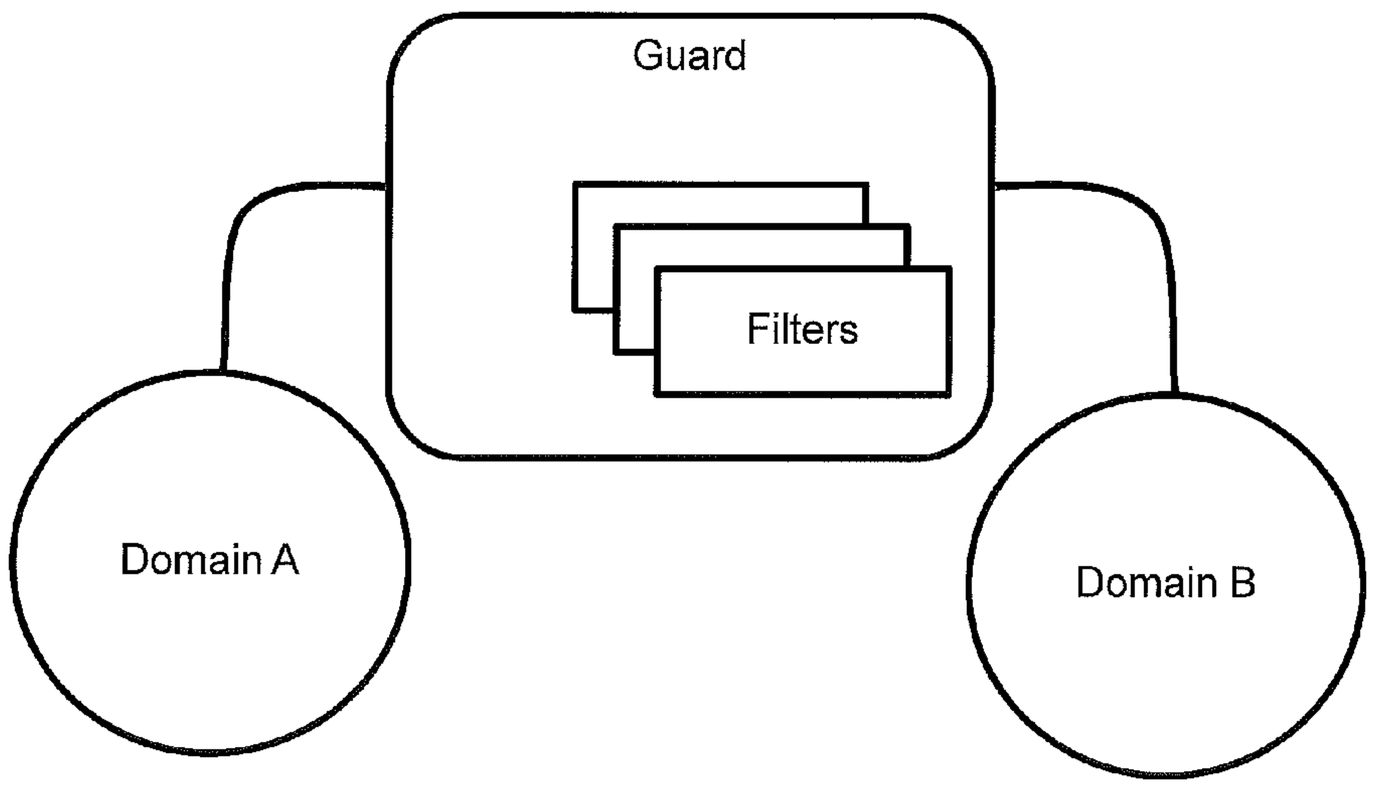


FIG. 2

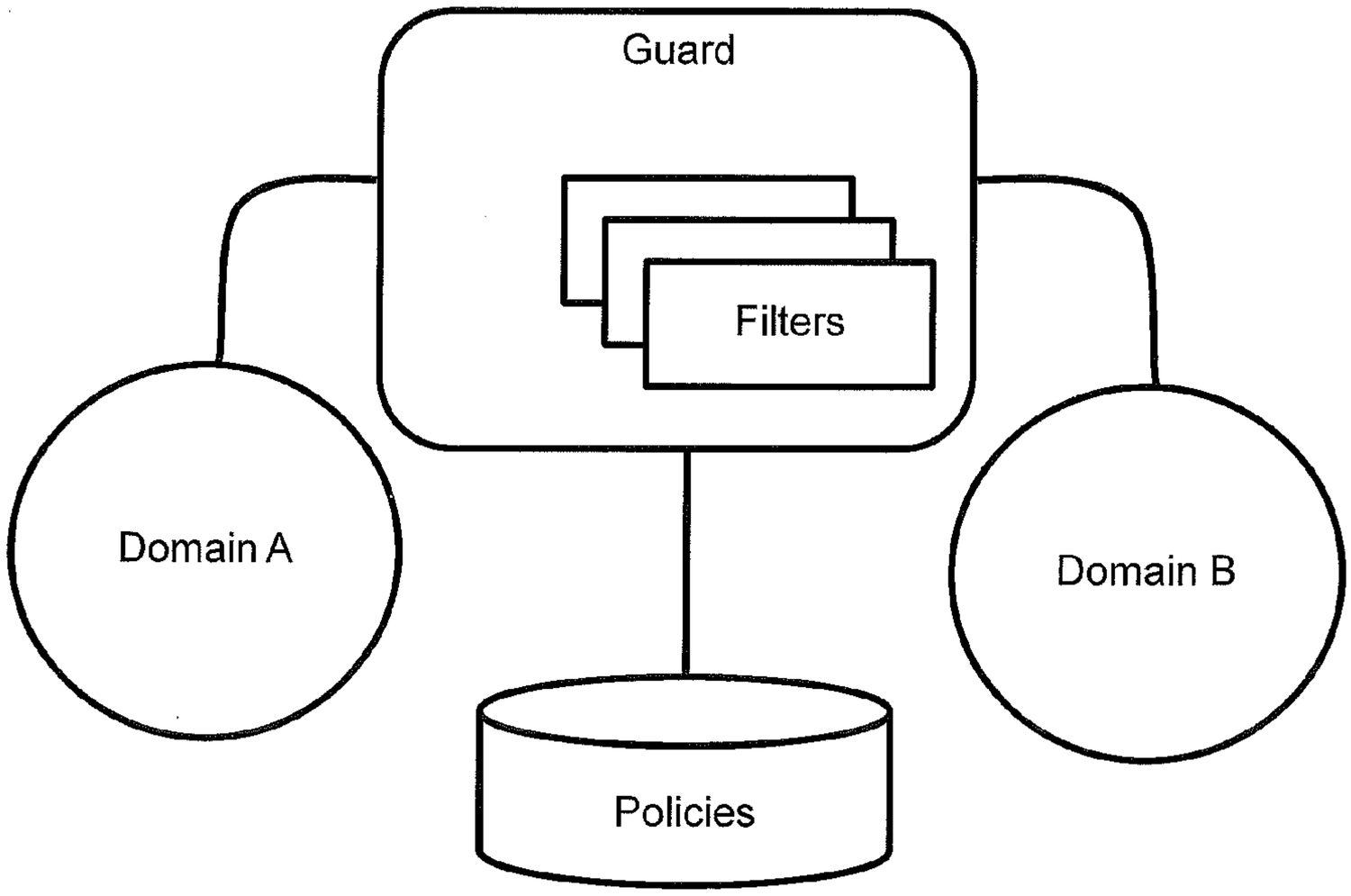


FIG. 3

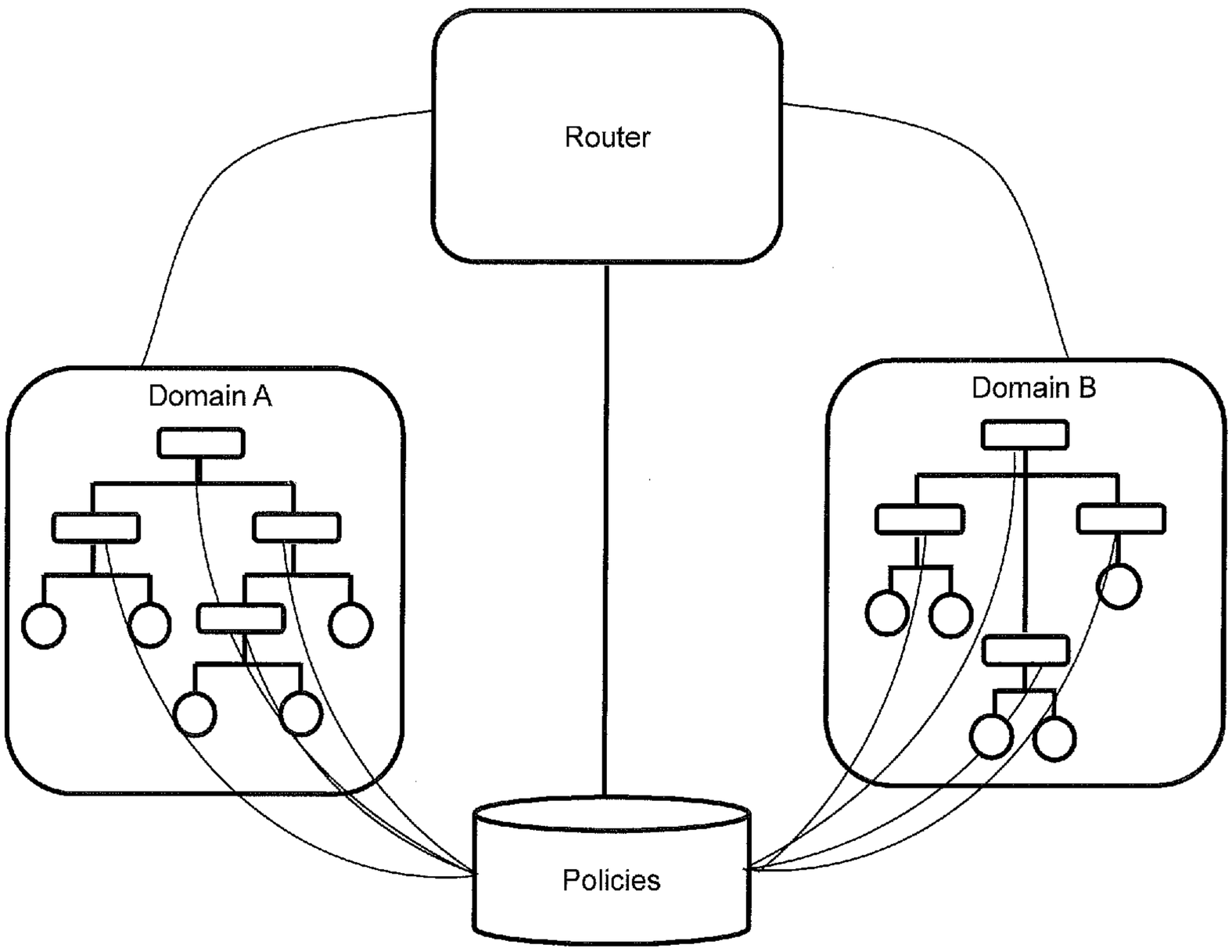


FIG. 4

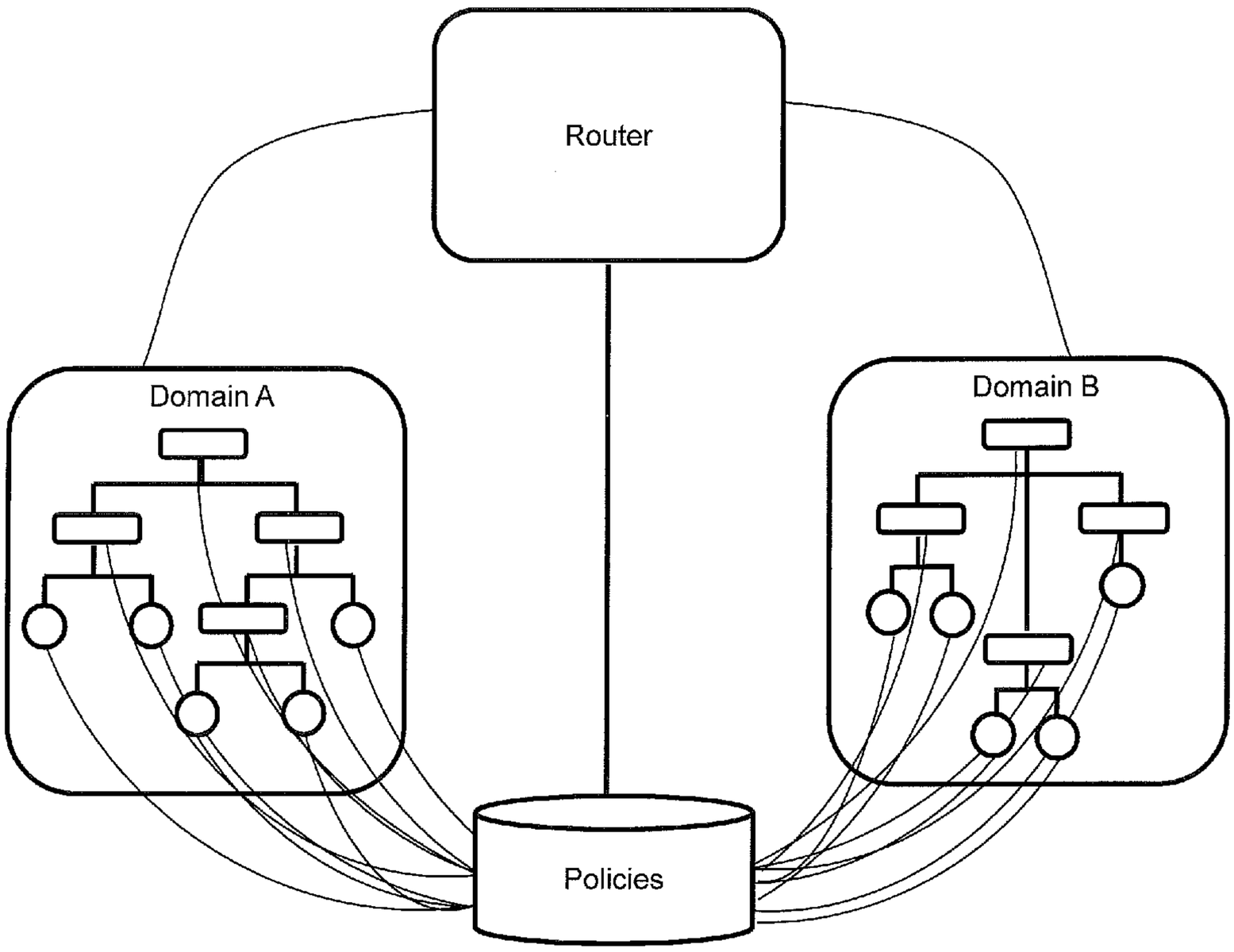


FIG. 5

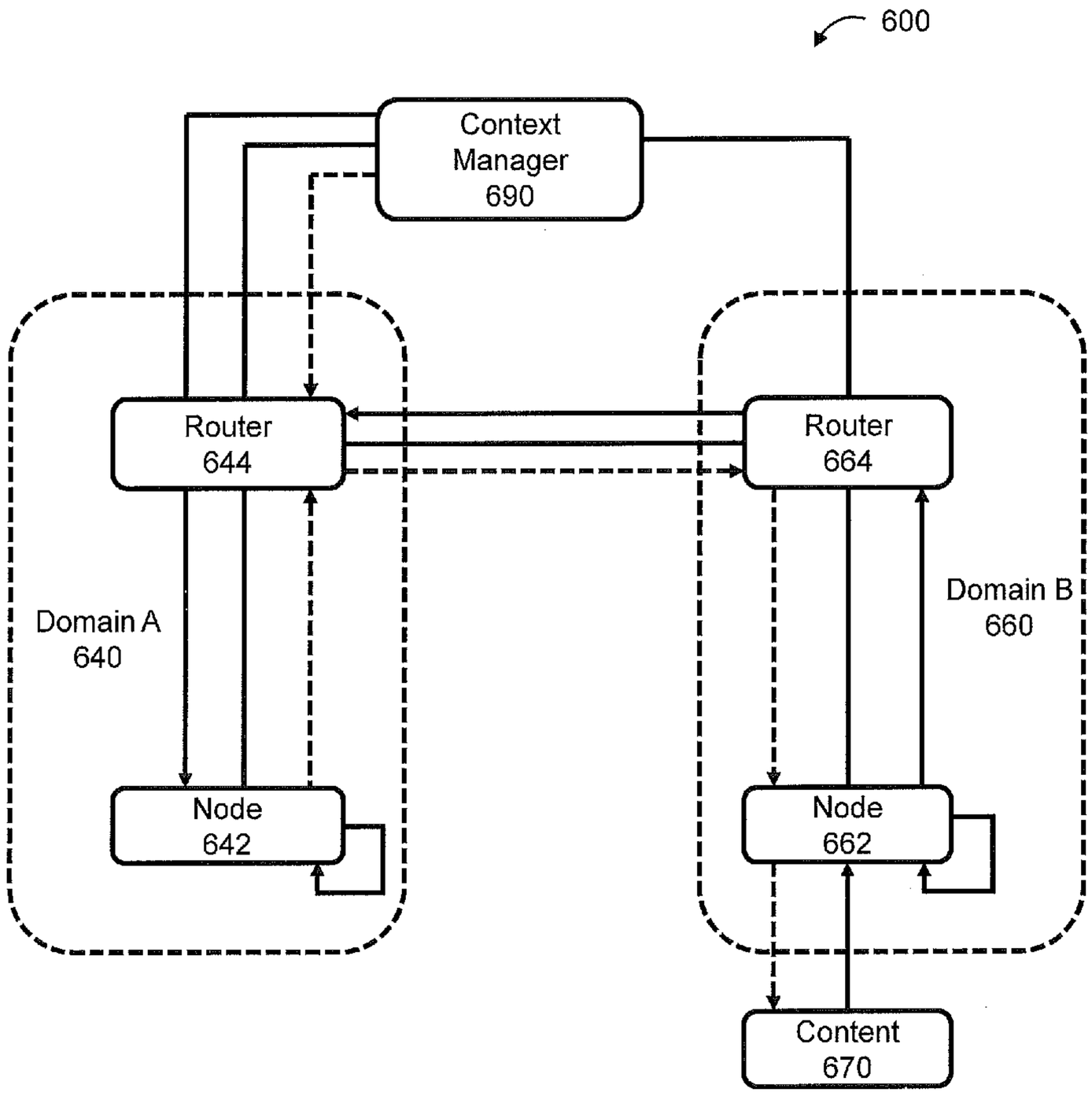

600
Context Manager 690
Router 644
Router 664
Domain A 640
Domain B 660
Node 642
Node 662
Content 670

FIG. 6

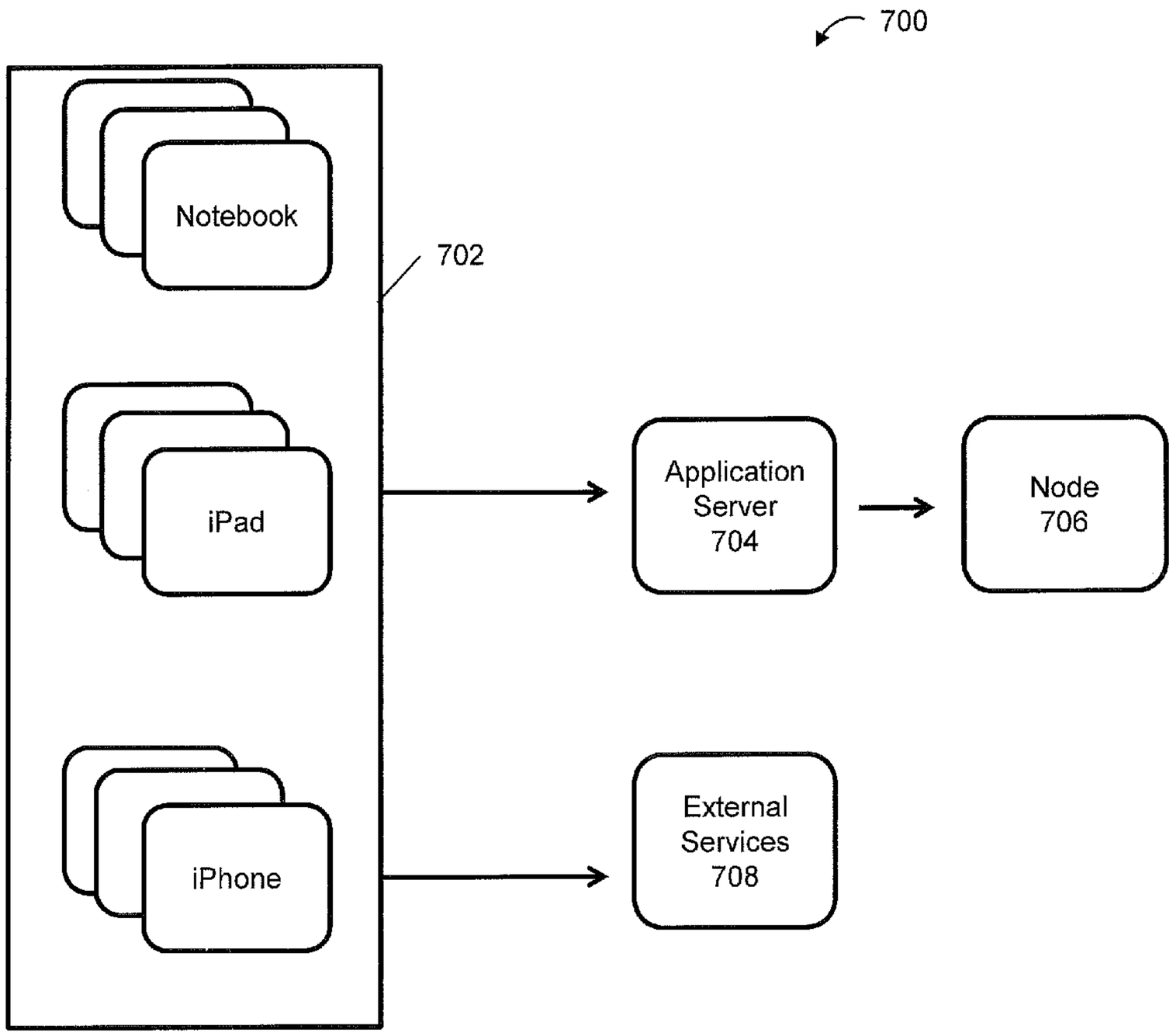

700
Notebook
702
iPad
Application Server 704
Node 706
iPhone
External Services 708

FIG. 7

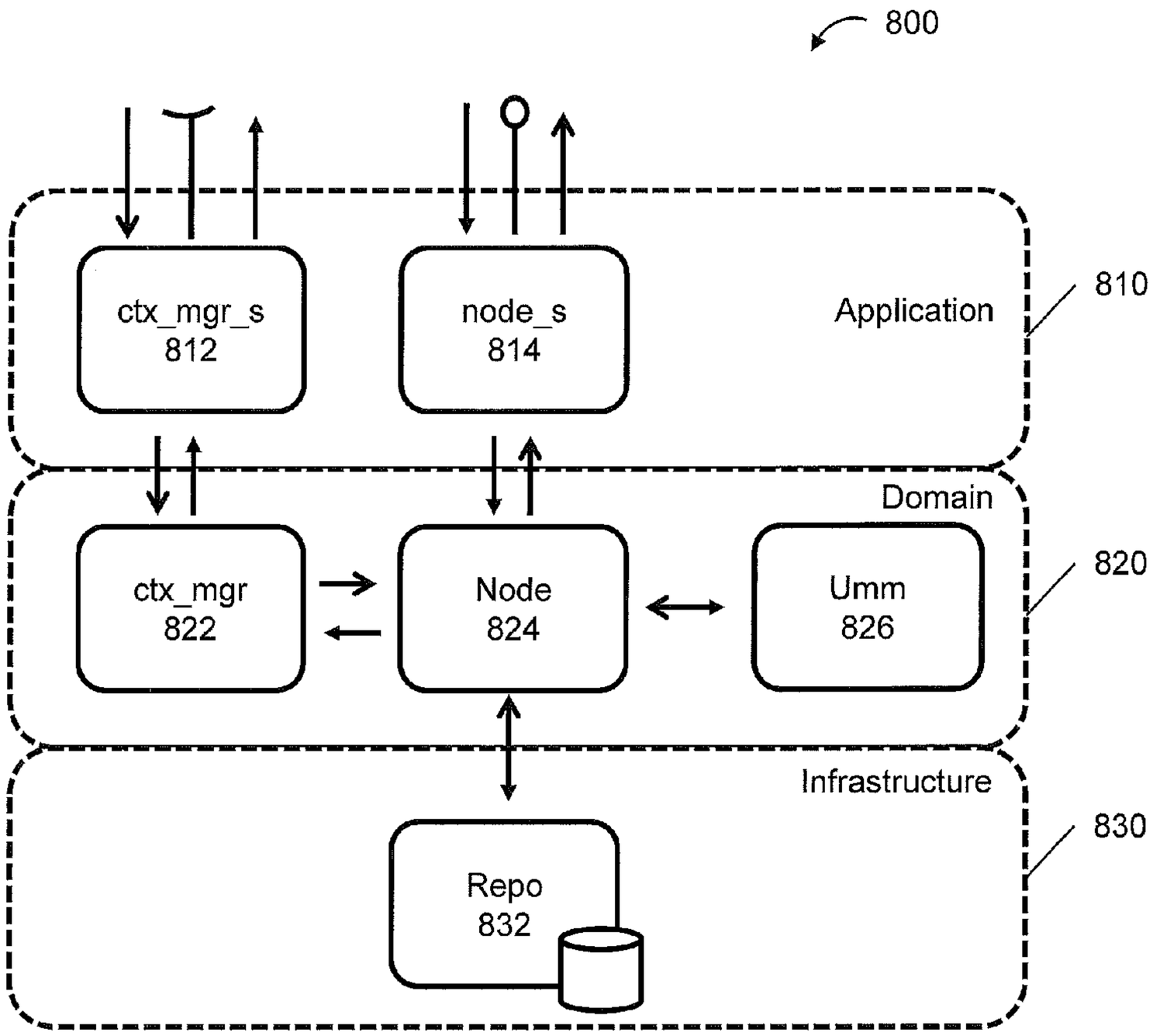

800
ctx_mgr_s
812
node_s
814
Application
810
Domain
ctx_mgr
822
Node
824
Umm
826
820
Infrastructure
Repo
832
830

FIG. 8

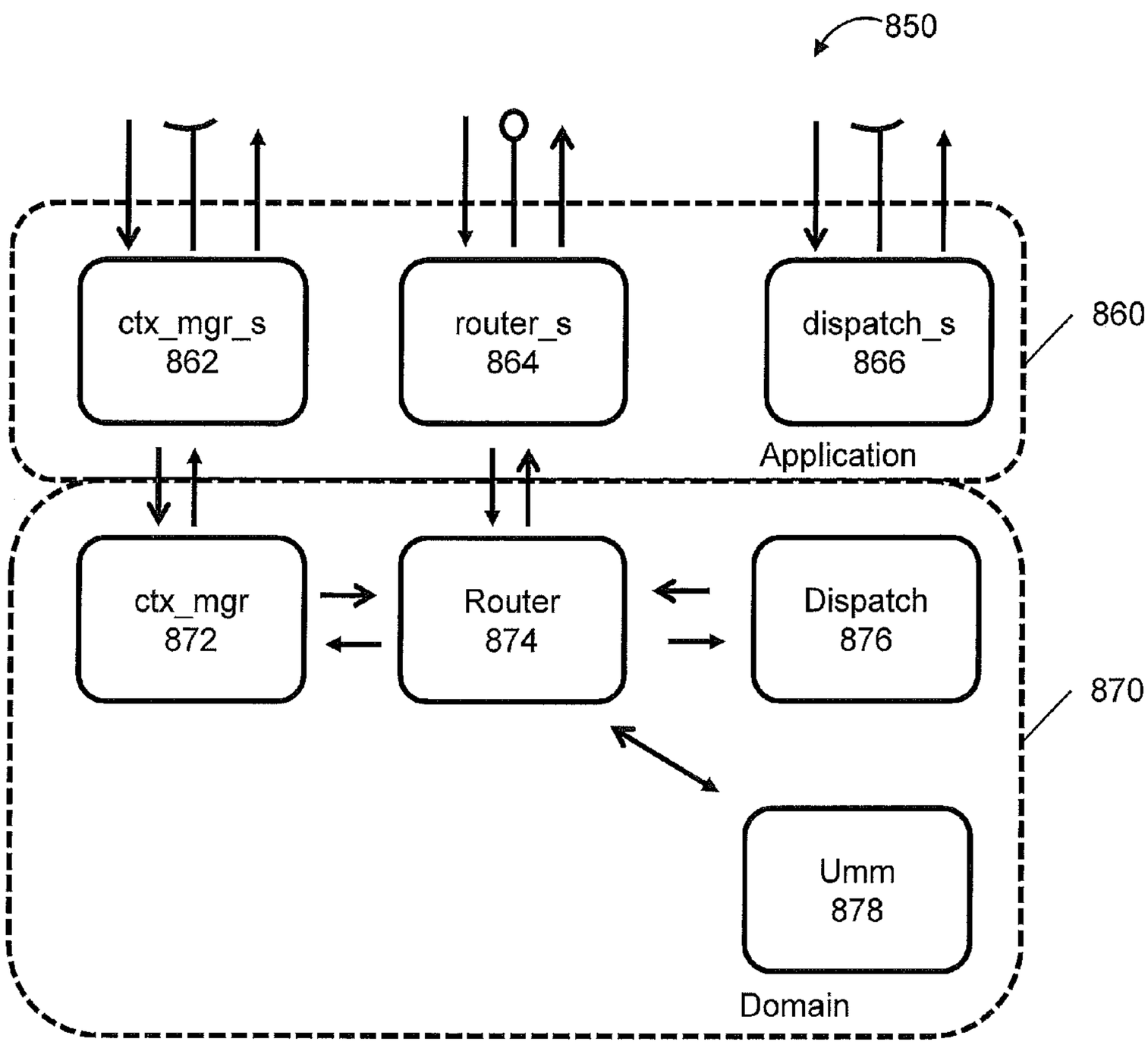

850
ctx_mgr_s
862
router_s
864
dispatch_s
866
860
Application
ctx_mgr
872
Router
874
Dispatch
876
870
Umm
878
Domain

FIG. 9

```
<index>
  <location>
    <name>The location name; a city name, for example</name>
    <lat>The location latitude</lat>
    <lon>The location longitude</lon>
    <about>Metadata about the location</about>
    <key>The detail data object name</key>
    <key>...</key>
    ....
  </location>
  <location>...</location>
  <location>...</location>
  ...
</index>
```

FIG. 10

```
<artifact>
  <policy-set>
  ...
  </policy-set>
  <data-object>
    <image type=".">
    ...
    </image>
  </data-object>
  ...
</artifact>
```

(a)

```
<artifact>
  <policy-set>
  ...
  </policy-set>
  <data-object>
    <shape type=".">
    ...
    </shape>
  </data-object>
  ...
</artifact>
```

(b)

```
<artifact>
  <policy-set>
  ...
  </policy-set>
  <data-object>
    <content type=".">
    ...
    </content>
  </data-object>
  ...
</artifact>
```

(c)

FIG. 11

```
...
<shape type="marker">
  <marker>
    <lat>...</lat>
    <lon>...</lon>
  </marker>
</shape>
...
```

(a)

```
...
<shape type="circle">
  <center>
    <lat>...</lat>
    <lon>...</lon>
  </center>
  </radius>...</radius>
</shape>
...
```

(b)

```
...
<shape type="polygon">
  <vertex>...</vertex>
  ...
</shape>
...
```

(c)

FIG. 12

Dimension	*Type*	*Required?*	*Domain A*	*Domain B*	*Domain C*
Affiliation	Set	Yes	tropic_thunder, gallant_entry	tropic_thunder, gallant_entry	tropic_thunder, curious_response
Sensitivity	Ordering	Yes	unclassified, secret, top_secret	unclassified secret, top_secret	unclassified, secret, top_secret
Category	Set	No	aqua, magenta, vermillion	alpha, beta, gamma	one, two, three
Organization	Set	Yes	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia
Device	Set	No	workstation, tablet, phone	workstation, phone	workstation, tablet

FIG. 13

Dimension	*Type*	*Required?*	*Domain A*	*Domain B*	*Domain C*
Affiliation	Set	Yes	tropic_thunder, gallant_entry	tropic_thunder, gallant_entry	tropic_thunder, curious_response
Clearance	Ordering	Yes	unclassified, secret, top_secret	unclassified secret, top_secret	unclassified, secret, top_secret
Category	Set	No	aqua, magenta, vermillion	alpha, beta, gamma	one, two, three
Organization	Set	Yes	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia

FIG. 14

```
policy_set {
  policy(:p1) {
    match :all
    rule(:mission_affiliation) { |x| x == :tropic_thunder }
    rule(:sensitivity) { |x| x == :top_secret }
  }

  policy(:p2) {
    include :p1
    match :all
    rule(:device) { |d| d == :workstation || d == :phone }
  }

  policy(:p3) {
    include :p1
    match :one
    rule(:category) { |c| c == :vermillion }
    rule(:organization) { |o| == :oceania }
  }
}
```

FIG. 15

```
typedef policy_set string;
typedef artifact string;

struct artifact_descriptor {
  policy_set policy_set;
  artifact artifact;
};

typedef sequence<artifact_descriptor> artifact_descriptor_list;
```

FIG. 16

```
enum status { unsecured, confidential, secret, top_secret };

struct link_status {
  string name;
  status status;
};

typedef sequence<link_status> link_status_list;

struct context {
  date date;
  link_status_list network_Status;
};
```

FIG. 17

```
exception error {
  string message;
};

exception client_error : error {};
exception server_error : error {};
exception unknown_response_error : error {};
```

FIG. 18

```
typedef string user_name;
typedef user_name subject;
typedef string key;

enum device { tablet, phone, workstation };

interface artifact_manager {
  artifact_descriptor get_artifact(in subject s, in device d, in key k) raises (error)
      ;
  artifact_descriptor_list get_all_artifacts(in subject s, in device d) raises (error)
      ;
};
```

FIG. 19

```
1 interface ContextManager {
2   context context() raises (NetworkError);
3 };
```

FIG. 20

```
1 enum activity { transmit };
2 typedef string policy;
3
4 [Constructor(in ContextManager contextManager);]
5 interface usage_management_mechanism {
6   bool can_execute(in policy p, in context c, in activity a);
7 };
```

FIG. 21

```
1 interface repository {
2   artifact_descriptor get_artifact(in key k);
3   artifact_descriptor_list get_all_artifacts();
4 };
```

FIG. 22

```
1 [Constructor(in string host);]
2 interface dispatcher {
3   artifact_descriptor_list dispatch(in user u, in device d, in key k) raises (error);
4 };
```

FIG. 23

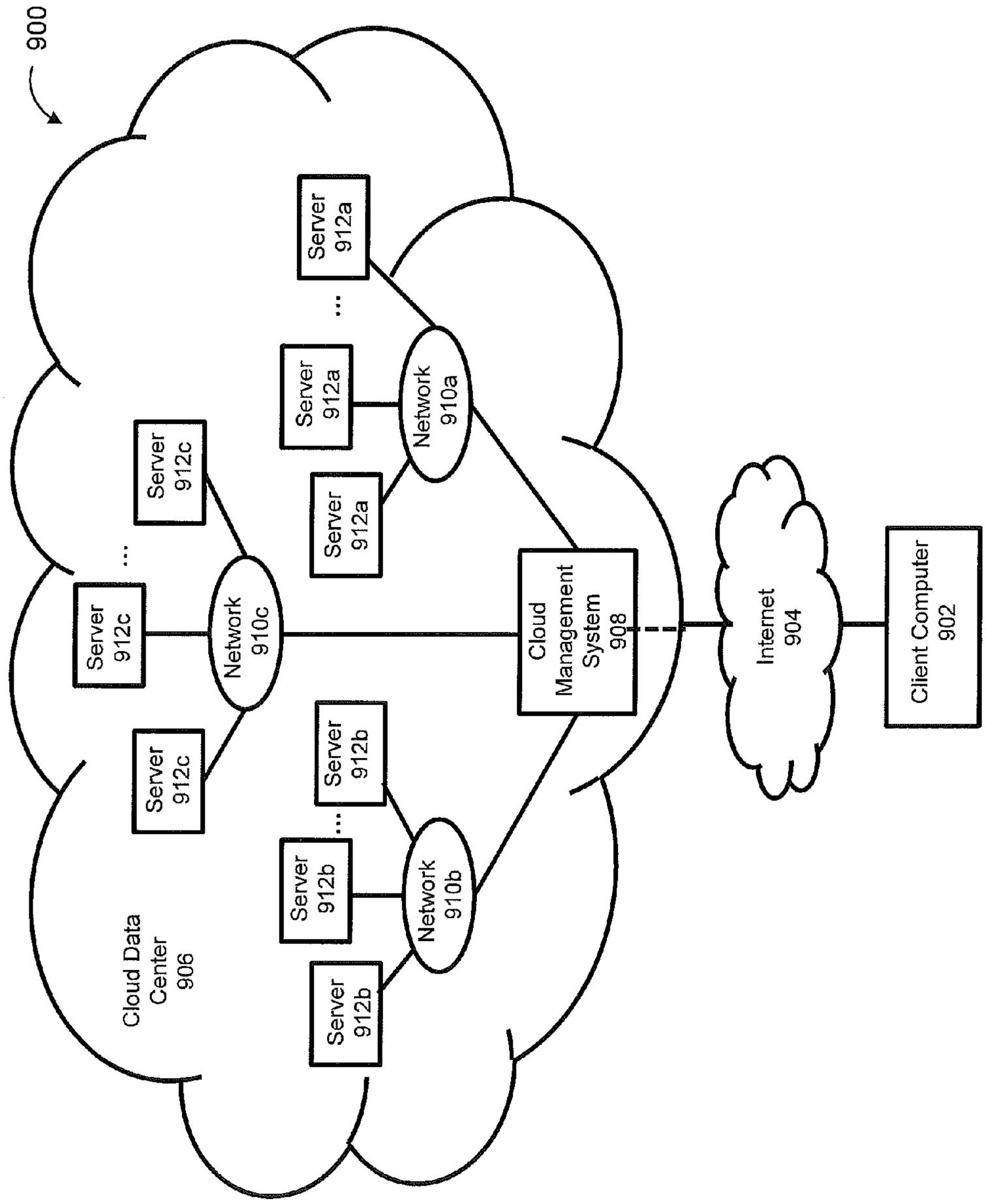

900
Server 912a
Server 912a
Server 912a
Network 910a
Server 912c
Server 912c
Server 912c
Network 910c
Server 912b
Server 912b
Server 912b
Network 910b
Cloud Data Center 906
Cloud Management System 908
Internet 904
Client Computer 902

FIG. 24

SYSTEM AND METHODS FOR USAGE MANAGEMENT IN MULTI-LEVEL SECURITY NETWORKS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to U.S. Provisional Application No. 61/639,162, filed Apr. 27, 2012, incorporated by reference in its entirety.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH

This invention was made with government supported under grant no. FA8750-10-C-0090 awarded by the Air Force Research Laboratory (AFRL). The United States Government has certain rights in the invention.

FIELD OF THE INVENTION

The invention relates generally to computing systems. More specifically, the invention relates to overlay networks that support the transfer of information between two or more incompatible security domains or levels of classification including usage management in cloud computing environments that partitions information in cross-domain networks.

BACKGROUND OF THE INVENTION

Current enterprise computing systems are too expensive, unreliable, and information dissemination procedures are slow. Current approaches to partitioning information in cross-domain scenarios are simply unable to migrate to cloud environments because of reliance on control of physical hardware to enforce information separation. The current approach of controlling information by controlling the underlying physical network—the traditional approach to securing information—does not scale into shared datacenters thereby risking exposure of sensitive data. The term "cross-domain" refers to the transfer of information between two or more incompatible security domains or levels of classification.

Typically, systems handling sensitive information use costly data partitioning schemes. Most of these kinds of systems are managed in-house rather than exploiting lower cost cloud-enabled services. Furthermore, many of these systems have large maintenance loads imposed on them as a result of internal infrastructural requirements like data and database management or systems administration. In many cases networks containing sensitive data are separated from other internal networks to enhance data security at the expense of productivity, leading to decreased working efficiencies and increased costs.

These kinds of large distributed systems suffer from a lack of stability and reliability as a direct result of their inflated provisioning and support costs. Simply put, the large cost and effort burden of these systems precludes the ability to implement the appropriate redundancy and fault tolerance in each and every system. Justifying the costs associated with standard reliability practices like diverse entry or geographically separated hot spares is more and more difficult to do unless forced by draconian legal policy or similarly dire business conditions.

Finally, the length of time between when a sensitive document or other type of data artifact is requested and when it can be delivered to a requester to view that artifact is prohibitively long. These kinds of sensitive artifacts, usually maintained on

partitioned networks or systems, require large amounts of review by specially trained reviewers prior to release to data requesters. In cases where acquisition of this data is under hard time constraints, like sudden market shifts or other unexpected conditional changes, this long review time can result in consequences ranging from financial losses to loss of life.

Federal, military, and healthcare computer systems are just a few prime examples of these kinds of problematic distributed systems, and demonstrate the difficulty inherent in implementing new technical solutions. New approaches to networking and information management present possible solutions to these kinds of problems by providing distributed information-centric approaches to data management and transfer.

Current policy-centric systems are being forced to move to cloud environments and incorporate much more open systems. Some of these environments are private or hybrid cloud systems. Private clouds include infrastructure that is completely run and operated by a single organization for use and provisioning. Hybrid clouds include a combination of private and public cloud systems.

Many organizations are poised to benefit from the migration of policy-centric systems to cloud environments including, for example, the United States National Security Agency (NSA) and the United States Department of Defense (DoD), both of whom have large installed bases of compartmentalized and classified data.

Cloud systems provide a variety of economic incentives for use, for example cost savings and flexibility. However, cloud computing systems have distinct disadvantages such as issues relating to trust and security as well as information sensitivity problems.

Current cross-domain models all use some kind of filter chaining mechanism to evaluate whether a given data item can be moved from a classified network to an unclassified network. Certain cross-domain models use filters explicitly as well as use a single point of security and enforcement, providing perimeter data security, but nothing else. In current system architectures, users are only allowed to exchange one type of information per domain. The physical instantiations of these models are locked by operational policy to a single classification level. Users cannot, for example, have "top secret" material on a network accredited for "secret" material. Finally, these models violate end-to-end principles in large service network design, centralizing intelligence rather than pushing that intelligence down to the ends of the system.

End-to-end principles are generally considered core to the development of extreme scale, distributed systems. Essentially, one of the key design decisions with respect to the early Internet was to move any significant processing to system end nodes, keeping the core of the network fast and simple. Known as the end-to-end principles, this design has served the Internet well, allowing it to scale to sizes unconceived when originally built.

Current cross-domain systems are placed at key routing points between sensitive networks. These locations are core to information transfer between systems and as a result violate the initial design principles upon which the Internet was founded. End-to-end principles need to be modified to support future networks, but nevertheless, current cross-domain systems still violate the basic ideas behind large, scalable networks by placing complex application-specific logic directly and only in the core of a given sensitive network.

There is a need for decentralized policy management capabilities, infrastructural reuse, the ability to integrate with cloud systems, and security in depth. Policy management needs to be decentralized and integrated within the fabric of

the system such that the system is both more secure and resilient as a result, better able to control information and operate under stressful conditions. Multi-tenancy can lower costs and increase reliability and is furthermore a common attribute of cloud systems. An appropriately secured system facilitates integration of computing resources into multi-tenant environments. The ability to handle multi-tenant environments and to reliably secure both data at rest and data in motion leads to computational environments deployable in cloud systems. Finally, systems must operate under all conditions, including when they are under attack or compromise and provide protection to sensitive data in depth. The invention is a system and methods that supports the timely delivery of secure, robust, and cost-effective cross-domain capabilities and enterprise services that enables the sharing of information across security domains.

SUMMARY OF THE INVENTION

Information-centric networking is a new approach to Internet-scale networks that shows promise with respect to decentralized, information-centric usage management, addressing scale and availability issues with current systems. Specifically, information-centric networks provide more efficient content management and supplies new capabilities for information security. Usage management refers to the ability to control how resources (data and services) are used across and within computing domains. Controlling how information is used becomes increasingly difficult as computing infrastructure becomes more distributed. However, the ability to share information between domains provides for powerful capabilities, as well as increased security risks.

According to the invention, usage management incorporates aspects of access control and digital rights management (DRM). Access control relates to the access rules defined in terms of relationships between a set of resources and a set of users. DRM relates to specifying and enforcing rules related to how the resource can be used. The invention integrates usage management with an information-centric network in either hierarchical or non-hierarchical (peer-to-peer) configurations.

In general, it takes extensive advantage of data locality, caches data aggressively, decouples information providers from consumers, and uses an information-centric perspective in network design. Information-centric networking provides higher information availability through better network resilience and implementing systems that more closely reflect today's use, focusing on heterogeneous systems with requirements ranging from mobile to static access. It is believed that the current Internet is not well suited to the way it is used today and that in order to efficiently support future use, the Internet needs to be fundamentally re-examined and perhaps, in some ways, re-implemented. However, different types of information-centric networks are not all synchronous.

The invention introduces the notion of usage management embedded in a delivery network itself. The invention considers the challenges and principles involved in the design of an open, inter-operable usage management framework that operates over this kind of environment including the application of well-known principles of system design and standards, research developments in the areas of usage control, policy languages design principles, digital rights management (DRM) systems, and interoperability towards the development of supporting frameworks.

The invention is directed to overlay networks that use usage policies for content management by dividing a given system into specific security domains which are governed by individual policies. This system fits into this proposed taxonomy as an a-type system as it has domains with single separating guards.

Information-centric networks present an opportunity to bring standards and theoretical solutions together into a new type of system providing unique and more powerful information management capabilities. The invention migrates these capabilities into information-centric networks.

Specifically, information-centric networks provide capabilities that traditional packetized networks cannot when it comes to managing the usage of information resources. The basic structure of packet networks facilitates simple and efficient data transfer, but is fundamentally based on certain design assumptions that render network-centric usage management difficult at best and impossible at worst. Information-centric networks, taking a very different approach to network design, are much more amenable to embedded content control based on their different design principles.

Current packet-based systems share three underlying design principles. Strict layering, in which upper layers only use services that exist in lower layers which in turn have no knowledge of upper layers, end-to-end arguments governing service placement, and limited runtime packet sizes. All three of these increase the difficulty in applying control over information transmitted through networks.

In Internet systems, switching and routing traditionally occur in the lower layers of the Open Systems Interconnection (OSI) model. These decisions are made based on a priori knowledge of a given network topology, by manual or programmatic configuration and are not impacted by transmitted content except in very high-end systems. In fact, access to application content occurs at much higher levels. As a result of strict service layering, the information needed to make content-sensitive routing decisions is simply not available without breaking layer encapsulation on these kinds of devices.

End-to-end arguments dictate where services should be placed in a network. Services like information distribution control that require access to application layer data should, following these principles, be deployed into the ends of a given network. In order to control information flow based on content, internal network nodes must be able to access and evaluate transmitted content. Policies associated with content can be arbitrarily large. As a result, they can exceed maximum packet sizes defined in packetized networks. Furthermore, as content sensitive networks must evaluate defined policies prior to routing content, any policy to be evaluated must be completely downloaded into a router and analyzed for suitability for transmission prior to any packet routing, leading to inevitable bottlenecks as content is queued behind the policy elements.

Content analysis of certain kinds of transmitted artifacts may not be possible without a holistic perspective either. For example, if an eXtensible Markup Language (XML) document is transmitted through a network, that document may very well have content in element n which is described in more detail in element n+2. Here, element n and element n+2, by themselves, are not sensitive. When combined however, they are. When transmitted, these elements would be in separate packets. For the sake of this example, it is assumed that these packets are built such that element n is in packet m, and element n+2 is in packet m+c, where c is some constant, and that packet m is assembled and transmitted from the source node at some time prior to packet m+c. In this scenario, packet m is passed through intervening nodes prior to packet m+c. Even nodes that maintain a history of transmitted content that may be able to determine that information in m is

sensitive when combined with information in m+c will be unable to undo the earlier transmission of packet m. In order to circumvent this problem, nodes need to hold packets for some time t to check for context. This may help solve the problem, as related information likely has some kind of intrinsic locality, but nevertheless the size of c can be still be relatively arbitrary. As a result, the size of t is impossible to set a priori.

This approach imposes possibly significant performance penalties as well. Information-centric networks are based on different primitives. Specifically, they are based on named data objects with strict name-data integrity, as well as other associated principles. This different abstraction makes policy evaluation and content binding simpler, as content can be bound either in-line to policy or via specific naming conventions. In these systems, once content is located by name, it is returned to the requester either via a predefined path mirroring the original request path or a variable response path. In either case however, all content and associated policy is available at each routing node in that return path, and can be evaluated for suitability of transmission. As a result of these fundamentally different underlying models, information-centric networks in the next-generation Internet enable usage management capabilities that are very problematic to implement and enforce in current Internet architectures.

Examining content at each network node or router point can certainly impact performance and extension availability. It is also important to establish that this kind of dynamic dispatch guarantees delivery along the most secure path needed. With respect to delivery, by selecting optimum paths at given network points the overall selected path will have the appropriate security characteristics outlined by any policy associated with delivered content.

In a given aggregate path between two points, if local decisions are made with respect to routing based on specific security criteria at interleaving points, the path as a whole will adhere to those security criteria. Essentially, this implies that it is possible to use a greedy algorithm with respect to security and routing and that the algorithm will yield an optimal security path. It is important to recognize that this is key to establishing a secure route between two specific points. Furthermore, in a given route, that route must be viewed temporally as well, in that each link may not be optimal when the delivered data element reaches a destination, but each link was optimal at the time it was selected, and by extension, when the aggregate path is reviewed, it would likewise be optimal with respect to time of traversal. Finally, local nodes may very well have knowledge about the local environment that cannot be known by a centralized routing authority. Allowing local routing decisions with respect to security can help take advantage of this locality.

The invention and its attributes and advantages may be further understood and appreciated with reference to the detailed description below of one contemplated embodiment, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The preferred embodiments of the invention will be described in conjunction with the appended drawings provided to illustrate and not to limit the invention, where like designations denote like elements, and in which:

FIG. **1** illustrates a table of usage management taxonomy according to one embodiment of the invention.

FIG. **2** illustrates a diagram of ø taxonomy according to one embodiment of the invention.

FIG. **3** illustrates a diagram of α taxonomy according to one embodiment of the invention.

FIG. **4** illustrates a diagram of β taxonomy according to one embodiment of the invention.

FIG. **5** illustrates a diagram of γ taxonomy according to one embodiment of the invention.

FIG. **6** illustrates a diagram of the workflow across domains according to one embodiment of the invention.

FIG. **7** illustrates a diagram of the overall system architecture according to one embodiment of the invention.

FIG. **8** illustrates a diagram of the node architecture according to one embodiment of the invention.

FIG. **9** illustrates a diagram of the router architecture according to one embodiment of the invention.

FIG. **10** illustrates a XML file relating to seed information for a network according to one embodiment of the invention.

FIG. **11** illustrates a XML file relating to image, shape, and content information according to one embodiment of the invention.

FIG. **12** illustrates a XML file relating to marker, circle, and polygon information according to one embodiment of the invention.

FIG. **13** illustrates a table of all possible attributes for usage management decisions according to one embodiment of the invention.

FIG. **14** illustrates a table of all possible attributes for usage management decisions specific to users according to one embodiment of the invention.

FIG. **15** illustrates a Domain Specific Language (DSL) file relating to policy domain specific language supporting a subset of XACML elements according to one embodiment of the invention.

FIG. **16** illustrates an Interface Description Language (IDL) file relating to key artifact data types according to one embodiment of the invention.

FIG. **17** illustrates an IDL file relating to key status data types according to one embodiment of the invention.

FIG. **18** illustrates an IDL file relating to key error data types according to one embodiment of the invention.

FIG. **19** illustrates an IDL file relating to the node interface according to one embodiment of the invention.

FIG. **20** illustrates an IDL file relating to the context manager interface according to one embodiment of the invention.

FIG. **21** illustrates an IDL file relating to the usage management mechanism interface according to one embodiment of the invention.

FIG. **22** illustrates an IDL file relating to the repository interface according to one embodiment of the invention.

FIG. **23** illustrates an IDL file relating to the dispatcher interface according to one embodiment of the invention.

FIG. **24** illustrates an example of a cloud computing system that may be used to implement the methods according to the invention.

DETAILED DESCRIPTION

A clear taxonomic organization of potential steps in approaching finer-grained policy based usage management helps in describing the difficulties inherent in developing potential solutions as well as aiding in planning system evolution over time. Here, four distinct types of integrated policy-centric usage management systems have been identified, as shown in FIG. **1**.

In this taxonomy, it is not required that systems pass through lower levels to reach higher ones. This taxonomy represents a continuum of integration of usage management controls. Systems can very well be designed to fit into higher

taxonomic categories without addressing lower categories. However, many of the supporting infrastructural services, like identification management or logging and tracing systems, are common between multiple levels. The taxonomy itself starts with the current state, integrating policy evaluation systems into the network fabric gradually, moving away from filters, adding policy evaluation into the routing fabric, and finally into the computational nodes. The invention provides robust cross-domain capabilities, helps mitigate risk, and contributes toward advancing the state of multi-level security environments.

The ø classification consists of systems with two distinct domains, separated by a filter-centric single guard. As shown in FIG. **2**, two domains are separated by a guard using filter chains. Generally one of the domains supports more sensitive information than the other, but that is not always the case. Classified information is commonly stored in compartments which are separated by clear need-to-know policies enforced by access lists and classification guides. These kinds of compartments contain information at similar levels of classification, but contain distinct informational elements that should not be combined.

In these kinds of systems, specific rules regarding information transfer and domain characterization are tightly bound to individual filter implementations. They are based on a priori knowledge of the domains the guard connects, and therefore are tightly coupled to those domains. Furthermore, the filter elements are standalone within the system, in this classification, not availing themselves of external resources. Rather, they examine information transiting through the filter based purely on the content of that information.

The set of filters that could be developed and deployed within the guard are unlimited. For example, a filter may be created that inspects and possibly redacts the sections within the document, rather than passing or not passing the entire document through the guard. Indeed, if even very limited processing capabilities are assumed within the guard, then this guard can be made as powerful as any solution for implementing a cross-domain solution. Thus, the computational power of the guard is not the issue. The real issues are the benefits that can be gained by distributing the capabilities intelligently within the networked environment as opposed to fixing them programmatically and topologically at the perimeter of a sensitive network.

The α overlay classification contains systems that have begun to integrate policy-centric usage management. Both policies and contexts are dynamically delivered to the system. The dynamic delivery of context and policy allows these kinds of systems more flexibility with policy evaluation. The α category begins to integrate policy-centric management rather than using strict content filtering. The term "context" refers to the environment in which a resource is used by a subject.

As shown in FIG. **3**, two domains exist, although any number of domains is contemplated. Domain specific information is required to be tightly coupled to the filter implementations. Separating the permissions, obligations, and other constraints from the filters and incorporating them into a specific separate policy entity frees the guard from this coupling and provides additional flexibility to the system. The guard can continue to use filters to process data. These filters, however, are now more generic and decoupled from the specific domains the guard manages. The choice of using a specific filtering model rather than some other kind of construct is a design detail level to implementers. That said, however, individual filters will be remarkably different and still need to understand the ontologies over which specific licenses are defined rather than specific content semantics.

The policy repository is critical to the implementation and differentiation of this taxonomy category. This repository can be implemented as a separate repository keyed into a data artifact's unique name, for example. It could also represent a policy sent in tandem with a data artifact in a data package. The policy repository may be implemented as some kind of external service, and as such, represents the first such external service explicitly used in this taxonomy. Other external services may well exist and be used to adjudicate information transfer decisions as well.

The β taxonomic category begins to integrate policy-centric processing with router elements in a given network as shown in FIG. **4**. Systems based on this model can also host multiple domains as a result of flexible policy-based content examination. Each domain hosts a network of some kind, though that hosted network could very well be a degenerate network of a single system. Each network hosted in a domain is hierarchical, with specific computational nodes embodied by workstations, tablet computers or mobile devices, and routing points embodied by routers or switches of some kind.

Usage management has started to penetrate into the routing fabric of the network by doing content evaluation at router points. Content-based switching networks have been successful in other domains, and such techniques can be used here to provide policy evaluation capabilities. Certain types of traffic are easier to evaluate than others however. For example, HyperText Transfer Protocol (HTTP) requests and responses are easier to examine than Transmission Control Protocol (TCP) packets. When examining TCP packets, systems generally require additional context to select an appropriate packet window (e.g. the number of packets cached for examination). HTTP traffic does not usually require this kind of flexibility.

Information-centric networks, due to their shift from previous network models, follow this same kind of pattern and make the information the focus instead of the transferred bits. This migration of policy evaluation into the routing fabric provides for enhanced data security and better network management, especially if part of a network is compromised. Now that policy decisions can be made at the router level in a given network, network security in depth is beginning to emerge rather than simple perimeter protection. This not only provides the ability for additional information protection, but also allows for different compartments holding information at different need-to-know levels to be created ad-hoc under different routing segments. In cases of network compromise, this kind of dynamic policy enforcement can also allow for quick node excision as well.

As shown in FIG. **5**, the γ compartment has integrated policy evaluation with compute and routing nodes. Here, policies can be evaluated against content at all network levels—nodes emitting requests, nodes fielding requests, and all routing elements in between. The policy repository is supplying services to all computational elements in both domains. This provides increased granularity with respect to data compartmentalization by integrating information security into each network element. At this point, the network can create compartments of single nodes, while previously in β level systems compartments could only be created under specific routing elements. At this level, systems can also provide services revoking data access based on policy evaluation decisions when needed.

Furthermore, individual node exclusion is possible as well. A β classified system can excise network elements under specific routers by dynamic policy application. Now, the

same functionality can exist in individual compute nodes. For example, if a networked device like a smart phone is compromised, that device can be removed from access quickly or used to supply mis-information.

The levels of the taxonomy vary primarily with respect to the inclusion of policy-based usage management and information-centric structure. ø type systems are not structured with information-centric use in mind, nor do they use policy-centric management. Conversely, γ type systems are both purely policy oriented and completely information-centric.

As systems move through the levels of the taxonomy they gradually move from one side of the spectrum to another. Information-centric structures, hierarchical or otherwise, gradually migrate into the network beginning with β systems. Policy orientation is injected into the architectures starting with α systems and moving into the network fabric in parallel with information-centric exploitation.

In these systems, policy-based management supplies distinct advantages over filter-centric information control. This kind of policy-centric usage management is more content specific than filters, more flexible, and is more expressive than filter-centric systems. If content is impacted by a dynamic context that is defined in terms of the content itself and a request is made in a particular environment, then only under certain specific environmental conditions is the requested content allowed to be accessed. The decision to pass the content to the requester is based upon characteristics of the content related to dynamic changes within the environment. A filter-centric solution contained within the ø level of the taxonomy is unable to change filter rules based on changes like new content or environmental alteration as a result of the static nature of the deployed filters. A policy based system, on the other hand, is able to express the content specific policy easily for more dynamic evaluation. The clear demarcation of data objects simplifies management of trusted procedures and managed objects. For example, if content contains information that can only be accessed for a specific time period, a static filter based on evaluating content only cannot determine that the information in the content is no longer appropriate for dissemination after that time period ends. That kind of evaluation requires meta-data associated with the content that specifically describes these time bounds as well as a dynamic contextual evaluator to determine when that window of access has closed.

Policy-centric systems are more flexible than filter-based counterparts. In a filter-based solution, the type of content that can be evaluated is tightly coupled to the filters installed. If a given piece of content is new to a given filter-centric solution, that content cannot be appropriately examined and must be submitted for human review. A policy-based system is designed to be more general. When based upon a common ontology, the evaluation system can be very general with respect to its evaluation of a given policy. A general policy engine can handle a great variety of different content as long as the policies associated with that content correspond to known domain ontologies. This generality leads to a greater amount of flexibility with respect to what can be expressed in a specific policy.

A filter is going to have a specific responsibility, like redacting sensitive words from a document. In order for that filter to redact those sensitive words, it must have access to some kind of list of what those sensitive words are. Since ø level systems use static filters, the filters can only be updated when the filter itself is updated. Now a policy-centric system on the other hand can have a policy associating sensitivity with various areas of content in a specific document. In this case, all the system must do is understand the sensitivity described in the policy associated with the content, and can then redact that content if needed. The ontology describing the areas of sensitivity will change more slowly than the content itself, leading to a more flexible maintainable system. This is of course a simple example solvable by creating a dynamic list. The specificity of the filters requires additional complexity in the filter system itself. The generality of the policy-centric system allows the complexity to be more clearly expressed and contained within the policy file.

While a filter can process content at specific perimeter points, its lack of reach into a given network fabric limits the power a given filter can actually have over transmitted content. A policy associated with content, when transmitted with that content, can reference much more than the semantics of the protected content. That policy can describe specifically, in detail, how that content can be used. Filters cannot exercise that level of control.

In a distributed system with multiple filter points, information distribution can be controlled via deployed filters at a relatively fine level of granularity. This kind of distribution control cannot influence the use of protected content. However, once that content is distributed, possessors are accorded full access.

Policy-enabled systems are not limited in this way. Policies, when coupled with policy evaluation tools, can exercise control not only over distribution and routing, but also over use of distributed content at endpoints. These advantages accrue in usage management systems as policy capabilities are propagated through the information-centric fabric. Some of these advantages, like expressiveness, appear simply by beginning to use policies instead of filters. The remaining two have more of an impact as additional policy-centric nodes combine to form a system suitable for cloud deployment, increasing their impact as they move from α to γ types of systems.

Information-centric integration exhibits clear advantages over single point perimeter systems as well. Specifically, information-centric systems are more partitionable than perimeter solutions, enable content throttling, provide capabilities for dynamic content control, and allow content to be more traceable.

Filter-based perimeter protection is typically deployed at strategic routing points on secure networks. These kinds of networks are designed with specific regions of enhanced sensitivity separated by cross domain management systems regulating information flows. While sensible from the perspective of each protected region as a secure domain, this design thinking begins to fall apart when exposed to the very real threat of the malicious insider.

Boundary-centric information flow control is impossible to realistically achieve when the actual boundaries between malicious actors and system users is constantly in flux. When a malicious actor can be anywhere within a system, actual boundaries are simply too dynamic to be realistically recognized. In order to surmount this fluid system posture, a security in depth mindset must be adopted.

Information-centric networks enable this kind of defense in depth via the possibility of partitioning. An information-centric system can partition the user space and by doing so decrease the attack surface available to a malicious insider. ø and α level systems based on perimeter filters cannot do this. Systems beginning at the β level provide the potential to create need-to-know cells of finer granularity up to γ type systems in which cells can be created at the level of specific nodes. These need-to-know cells serve to help quarantine

possible intrusion into the sensitive distribution fabric if that fabric is compromised by helping isolate that system failure within a compromised cell.

As an example, a hypothetical system with nine nodes connected along a single data plane within a secure network can be compromised. With perimeter defenses, if one of the nine nodes is compromised, a malicious actor can potentially begin to monitor communications traffic between all network nodes, effectively compromising the entire network. In this same network, if designers partition the system into three cells of three nodes, a similar intrusion in one of those cells will effectively only compromise that cell, leaving the other two cells unaffected. This decrease in possible targets for compromise effectively decreased the network attack surface from any give node by ⅔, correspondingly increasing the security posture of the system.

Perimeter located filter systems only have the opportunity to control sensitive traffic at that perimeter boundary. Information located in repositories behind that boundary is not subject to control if it is retrieved by an agent also ensconced behind that same system boundary. Granted, control can be exerted at the repository level, but in a system with more than one repository, this is of limited impact.

A partitioned cell-oriented system, on the other hand, provides greater opportunity for information monitoring and control. The partitions provide additional potential control points that requests must cross in order to access needed information. Furthermore, less random cell design provides the capability to unify repositories, providing tight control of information dissemination.

The hypothetical nine-node system, for example, provides no control over information dispatched from one of the contained nodes to other contained nodes in its initial design form. There are no control points within that nine-node network at which to monitor and control information flow. Partitioning that space into three three-node cells provides at least one potential control point for all inter-cell requests at which information flow can be monitored. In cases where a malicious insider is actively collecting and hoarding data for exfiltration, these additional control points give administrators the ability to automatically throttle the rate at which sensitive material can be accessed by users to increase the cost of data collection and increase the likelihood of agent discovery.

Singular perimeter solutions due to their lack of internal control points also forgo the ability to provide dynamic content control. Once information has traversed a given perimeter access point, it is no longer under the control of that point and can no longer be retrieved, accessed, monitored, or modified. Solutions with internal control points can provide the ability to continually monitor and control disseminated information.

Within a given information-centric system, depending on that system structure, data can be more rigorously controlled. β and γ systems provide the ability to dynamically change information access via contextual changes at a finer grained level than perimeter solutions. γ systems can in fact provide the ability to retract information access on a per request basis. This kind of control is especially useful in situations where external partners may temporarily need access to sensitive information for a specific short period of time, say during some kind of joint exercise or activity. γ systems can provide that access only during the window of operation, and retract that access when that window closes.

Access rules in policies can describe the general access to data objects based on specific individual project context. This project context could then be embodied in attributes associated with the user to authorize specific actions over objects. In this way, a subject that has been granted access to information from project A can be dynamically denied access to content from project B, when projects A and B are mutually exclusive.

The singular location of perimeter filter solutions also precludes easy information traceability. Data requests within a given network sans internal controls are more difficult to trace than an information-centric solution with a partitioned cell structure that is tailored to the specific information requested such as XML databases or semantic web content. The partitioned information-centric system requires requests to traverse multiple routing nodes at which request and response content can be examined and stored for later analysis and visualization. Perimeter solutions without this kind of structure cannot monitor flows at this finer-grained level.

The advantages of information-centric systems over single perimeter points gradually build as information-centricity permeates any given system. Some abilities, like content centric access repudiation, can only occur at the γ level. Others, like traceability or throttling, become more effective as a system architecture traverses from lower to higher levels of capability within the proposed taxonomy.

Information-centric networks provide new ways to secure information. This kind of repeated content analysis, enabled by information-centric computation, can potentially delay information delivery unacceptably. Information integrity can also be damaged using some possible approaches. The specific impacts on availability and integrity of these increased confidentiality mechanisms are vital to understand when selecting between multiple options. For example, removing information from content prior to transmittal over unsecured network paths certainly protects that removed content, but destroys the integrity of the transmitted information. Likewise, constant encryption and decryption of data to enable repeated examination of transmitted content certainly has a negative performance impact.

The invention separates content management from physical communication networks to enable network infrastructure virtualization and multi-tenancy. According to the invention, overlay routers can in fact use licenses bundled alongside content to modify transmitted content based on dynamic network conditions. Running on a single host over HTTP, the invention simulates two content domains and communication between them. The communication link has uncertain security state and changes over time. Although the invention currently runs on a single host with varying ports, it could easily run on multiple hosts as well. The current single host configuration is simply to simplify system startup and shutdown.

License bundles are hosted on the file system, although they could be hosted in any other data store. These artifacts are currently XML. They are stored in a directory, and the license file has a “lic” extension while the content file has an “xml” extension. Both the content and the license files have the name of the directory in which they reside. For example, if the directory is named “test”, the license file is named “test.lic” and the content file “test.xml”. In this context, the directory is the content bundle. The license and content files are simply documents and port to document-centric storage systems easily. However, the files can certainly be stored in traditional relational databases as well.

As shown in FIG. **6**, the system **600** itself has two domains, “Domain A” **640** and “Domain B” **660**. Each domain consists of a client node and a content router node. Specifically, Domain A includes node **642** and router **644** and Domain B **660** includes node **662** and router **664**. A request is initially served to client node **662**. If the client node **662** has the requested content, the content **670** is returned. If client node

662 does not contain the requested content, the node 662 forwards that request to the affiliated content router 664. The content router 664 sends the request to all the content routers of which it is aware, here router 644. Router 644 will then query associated client node 642 for content. If the requested content is in fact found, it will be returned to the original requesting router 664 and then to the requesting node 662. If the content is not found, a "not found" error message such as a HTTP status 404 code is returned to requesting router 664 and node 662.

All router-to-router content traffic is modified based on security conditions. A context manager 690 maintains metadata regarding network paths. A policy-centric usage management decision is executed based on the known state of communication infrastructure. If a given network path is only cleared for data of a certain sensitivity level, a transmitting router will remove all license information and content that is associated with higher sensitivities, and then transmit only information at an appropriate sensitivity level over the link.

The system is current configured to use ports 4567 through 4571. All content requests are via HTTP GET. Link status can be changed via HTTP POST and the CURL command is used to access the network. A simple information-centric network for usage managed content over HTTP is implemented and may be easily extensible to Secured HTTP (HTTPS). Changes in the context of the network dynamically change the format of transmitted content. The network successfully filtered content based on policies and dynamic network conditions. The system also delivered both arbitrary content and policies within a single document, and it was not prohibitively difficult to extract either data or policies. Extending the system 600 from a single host to a fully distributed network is feasible.

At this point the system is a distributed content network distributed across multiple nodes and domains providing cross-domain managed data access. This network consists of clients accessing information through a user interface subsystem that accesses data from external sources and a distributed cross-domain information network. Queries are submitted through a client, to an application server, then to external services and information nodes.

The unique strength of this system is enabling dynamic distributed content control. This includes information retraction, redaction, protection, and secure routing. Information retraction involves quickly denying access to sensitive data. Redaction addresses simple data removal, while protection would operationally involve applying encryption layers of increasing strength based on operational demands. Finally, secure routing provides the ability to send data over a more secure link if such a link is available and required.

In this system, information retraction involves changing the execution context such that access for a given user, perhaps even on a specific device, is removed. This context then propagates through the information network and attached clients. This is useful when a given user is suddenly considered compromised and can no longer be allowed access to sensitive information. Likewise, a specific user's system may likewise be compromised and be forbidden access to specific information.

Information redaction is generally used when a user simply does not have authorization for a specific section of content, generally within a larger document. In these cases, that information and related policy metadata are simply removed from any query responses. Likewise, information protection also addresses specific subsections of information in a larger document, but unlike redaction, a user is in these cases authorized to access information, but one of the links over which the information must travel is not authorized to transmit specific sensitive information. In these cases, that information can be encrypted with appropriately strong encryption to allow for more secure information transmission.

Finally, secure routing use directly addresses the ability to select communication links based on information content. In these situations, a network has more than one path over which to return content. Furthermore, these multiple paths have different characteristics providing different levels of service. The system, based on rules contained in a policy and the current context can then select communication links of different security levels when returning content. Likewise, the content network must: (1) support and distribute queries for available content based on submitted constraints including artifact key, (2) support and distribute queries for specific content based on key, (3) evaluate returned content for suitability for transmission to a requesting node at each transmission step, (4) support partitioning into multiple domains, (5) allow for dynamic information distribution at network start, (6) collect experimental metrics for evaluation, and (7) be distributed across multiple nodes.

FIG. 7 illustrates an overall system architecture 700 that includes a content network 702, application server 704, node 706, and external data source services 708. Specifically, the system architecture 700 according to the invention consists of a markup language—HTML 5 based user interface subsystem. The user interface layer displays maps and associated metadata to users based on submitted geo-location information and supports two different mobile profiles (tablet and telephone) and a single workstation profile. HTML 5 media queries are used for end device detection, allowing developers to format information differently for the three profiles and thereby facilitating usability. External data sources 708 are any data programming interface offered by a third party, for example, Google Maps to define, download, display, and format maps. Finally, the content network 702 exists and is configurable either as a hierarchical network or a non-hierarchical network containing geo-tagged information at various sensitivity levels. The content network 702 can be configured arbitrarily, enabling the creation of a virtually unlimited number of different information domains. The client systems layer is replaced with a command-line interface and external services are not accessed, but a typical deployment operationally would have these elements.

The user interface subsystem processes requests and returns information from both the external data source 708 and the content network 702 based on those requests. In one embodiment, Ruby on Rails (RoR) using standard RoR configuration conventions running on top of Ruby 1.9.*. is used along with Rake for deployment, Gem for component installation, Bundler to maintain consistent application dependency state, and Ruby Version Manager (RVM) to manage Ruby virtual machine versions. It is contemplated that HTML 5 interface elements are defined using a stylesheet language (SASS) and a HTML Abstraction Markup Language (HAML).

Operationally, typical system use involves query submission, usage management rectification, and result display. Two distinct types of queries exist: (1) an initial query for a map of a specific location, generally triggered by entering some kind of geo-location parameters (though potentially using device-generated location information, allowing automatic map alignment with a user's current location) and (2) a query for specific sensitive information. Initial queries have two distinct sub-queries, one of map information and another of the content network to see what data is available. All content is usage managed to ensure that mashed information is consis-

tent from a data sensitivity perspective prior to display to the user. No information is cached within the interface subsystem.

The content network can be configured to run as an HTTP overlay system using HTTP routers and nodes or in a peer-to-peer configuration. In either case, queries can be submitted to the network from any one of the constituent nodes. Routers do not store data, but focus solely on routing queries through a hierarchical network. After initial submission, queries propagate throughout the network based on user-submitted search parameters. The content network physically runs on nodes.

According to the invention, the common functional flow is built around responding to content queries with information of appropriate sensitivity for a given query context. In general, systems are designed with a layered perspective, with an application layer fielding initial requests, a protocol-agnostic domain layer that manages query responses, and an infrastructure layer that contains specific required libraries and other technical artifacts. In these systems, the application layer handles HTTP protocol issues, translating requests from the lingua franca of HTTP into the domain language reflected in the domain layer. The infrastructure layer consists of various data management technologies called upon by the domain layer when needed.

FIG. **8** and FIG. **9** highlight communication ordering within components in a hierarchical content network and also illustrate the functional components within the system. Specifically, FIG. **8** illustrates a diagram of the node architecture **800** and FIG. **9** illustrates a diagram of the router architecture **850**.

Turning to the node architecture **800** of FIG. **8**, requests come in through the application layer **810** and are then handed off for processing to the domain layer **820**.

The domain layer **820** retrieves the current context and is responsible for data responses that are managed according to the current environmental context. The primary components in the application layer **810** of the node architecture **800** are small adapters intended to translate between HTTP protocols and domain components.

The application layer **810** includes a Context Manager Client Service (ctx_mgr_s) component **812** and a Node Service (node_s) component **814**. The domain layer **820** includes a Context Manager (ctx_mgr) component **822**, a Node (node) component **824**, and a Usage Management Mechanism (umm) component **826**. The Context Manager Client Service (ctx_mgr_s) component **812** is an adapter between the Context Manager (ctx_mgr) component **822** of the domain layer **820** and the external context service. The context manager component **822** manages information associated with a context. The Node (node) component **824** provides a Representational State Transfer (REST) interface to external clients. All content requests are initially sent to a known Node Service (node_s) component **814**. This is essentially the external interface to a given content network. A content network generally contains many distinct nodes as well. Node (node) component **824** contains all logic needed to process and respond to information requests. Specifically, nodes manage requests, responses, context evaluation, and usage management mechanism application. The Usage Management Mechanism (umm) component **826** applies rules grouped into policies against a known context to determine the acceptability of an intended action. Additionally, it indicates whether or not that action can proceed. The Node (node) component **824** communicates with an infrastructure layer **830** that includes an Information and Policy Repository (repo) component **832**. The Information and Policy Repository (repo) component **832** is unique to nodes and includes information and policy repositories that contain specific content, organized by key, and associated policies.

Turning to the router architecture **850** of FIG. **9**, requests come in through the application layer **860** and are then handed off for processing to the domain layer **870**. The domain layer **870** retrieves the current context and is responsible for query dispatch that is managed according to the current environmental context. Similar to the node architecture **800**, the router architecture **850** includes primary components in the application layer **860** that are small adapters intended to translate between HTTP protocols and domain components.

The application layer **860** includes a Context Manager Client Service (ctx_mgr_s) component **862**, a Router Service (router_s) component **864**, and a Dispatch Service (dispatch_s) component **866**. The domain layer **870** includes a Context Manager (ctx_mgr) component **872**, a Router (router) component **874**, a Dispatcher (dispatch) component **876**, and a Usage Management Mechanism (umm) component **878**. The Context Manager Client Service (ctx_mgr_s) component **862** is an adapter between the Context Manager (ctx_mgr) component **872** and the external context service. The Router Service (router_s) component **864** is essentially a customized HTTP router that dispatches content requests and responses through a hierarchical content network in accordance with established policies and the current environmental context. The Router Service (router_s) component **864** communicates with the Router (router) component **874** of the domain layer **870** and the Router (router) component **874** manages the distribution of information requests and responses, managing information dispersal throughout a content network in accordance with context and policy. The Dispatch Service (dispatch_s) component **866** dispatches information requests to known nodes based on known policies and context. The Dispatcher (dispatch) component **876** sends requests to known routers or nodes in the larger context network. The Usage Management Mechanism (umm) component **878** applies rules grouped into policies against a known context to determine the acceptability of an intended action and indicates whether or not that action can proceed.

The same components are used to assemble non-hierarchical networks, in which nodes have both content storage and policy storage as well as request response and dispatching responsibilities. It is contemplated that context management and usage management components are shared between all types of content networks as well as all types of component systems within those networks. Non-hierarchical nodes and hierarchical routers and nodes all need these kinds of services.

Information-centric networks are generalized constructs supplying the ability to manage distributed content more effectively. Users must control information security and privacy in a more granular way when data is arbitrarily combined. In order to do this effectively however, a simple protocol must be defined that allows connected systems to determine what kind of information is available.

A variety of approaches can be used with respect to information storage in these kinds of networks. In many ways, they exhibit behavior very similar to file systems. In an information-centric network, rather than asking for content via some kind of address, like a uniform resource locator (URL), a specific non-ambiguous name is used. This is very similar to how content management systems and web caches work today. These kinds of systems treat a URL as a name rather than an address, returning a cached image of the requested content rather than the content actually pointed to by the URL. This requires that consumers and caching agents rec-

ognize and manage the possibility of stale data, but that risk is generally worth the performance gain.

Information-centric networks can similarly optimize various aspects of content retrieval, returning the most local, highest quality, or most reliable data item, for example. In this content network, metadata is associated with specific locations as well as the locations themselves. Rather than optimizing with regard to location or quality, this network optimizes security posture. In order to do so, a simple data discovery protocol is in place so clients can discover what data is available.

Two different models support content access in this kind of network. The first, referred to herein as the "Cat Model", mimics typical file system interaction on unix-centric computers. The second, referred to herein as the "Index Model", acts more like a typical website, with a central index providing available options. Both models are can manage hierarchical content, a requirement for managing large volumes of information.

With the Cat Model, a user has read access to the network via a set of related commands. File systems follow a model where the available contents can be listed, access specific details of the contents, and then access individual content items themselves. In UNIX and unix inspired systems, these actions correspond to ls, for directory listings, and programs like cat, to allow access to specific individual content. File details are exposed by options on the ls command. Command-line access to a content network is also certainly feasible.

In the content network, the ls command traverses the network returning information describing contents based on the current security context. This context consists of the environment, the resource requested, and the subject requesting the resource. For example, a user with access to a content network via some kind of shell may list network content from a device at a given physical and network location and receive content listing A, while executing a listing from a different device from the same locations may generate content listing B, which can be significantly different from A based on contextual changes.

Another problem that arises with listing network contents is the fundamentally different nature of listing a relatively small directory on a local computer as opposed to the contents of a geographically dispersed network. The latency involved when reading this kind of local directory is small, and the number of elements to list is tractable. Unbounded networks like these information-centric networks do not support these kinds of assumptions. The time required to list the available contents on a dispersed content network can be significant.

A cat-like command on a content network suffers from similar problems. As content within an artifact can be marked with different sensitivity, displayed artifact content can change based on context as well. Likewise, large artifacts can take significant time to display on devices because of content dispersion issues.

The Index Model is commonly used in world-wide-web systems both large and small. With the Index Model, a small index file that lists available content on the network. This index could be associated with a policy and marked for sensitivity, and could contain links to content as well as metadata describing that content. This index essentially serves the function of the ls command in the Cat Model. Selecting a link from an index via a network client would then serve as the Cat Model's cat command. Similar issues with respect to network dispersion exist with showing the contents of artifacts in both models, and the index contents can seem to change with respect to changing context, as they are also associated with policy sets describing the use of content. Both models can also be optimized for project-centric content viewing or to show indicators with respect to expected content retrieval latency.

Organizationally, any kind of informational hierarchy within the network needs to be based on the semantics of referenced content rather than external factors. Information-centric networks use keys to locate content rather than addresses, so this hierarchical name would in fact be such a key rather than an address for the content.

Latency effects and content surprise are characteristics of the underlying content network rather than a specific interface approach.

An initial index object that contains location information and associated metadata seeds the network. This information is classed according to sensitivity and consists of names and latitude/longitude coordinates contained in an XML file, similar to that shown in FIG. **10**.

Any of these XML elements can be marked with an attribute, policy-set, which is the name of a policy set contained in the associated policy file. It is contained as an artifact with an associated policy set. Detail data objects are arbitrary XML documents that support the policy attribute. Image, shape, and content or text information are all supported as shown in FIG. **11**. Each different type is ensconced within an XML element corresponding to the type of data contained, and are delivered in a single XML document with associated policy sets. Specifically, FIG. **11**(*a*) illustrates a XML file relating to image information, FIG. **11**(*b*) illustrates a XML file relating to shape information, and FIG. **11**(*c*) illustrates a XML file relating to content information.

Data-objects can be associated with policies contained in the policy-set element. Each policy-set element can contain zero or more policies. Sections within the content element can also be associated with policy sets, and currently type can be either xml or txt. A shape can only be associated with a policy set from the shape element itself. Properties of a shape cannot be associated with a policy set individually. Shape types include marker, circle, and polygon, as shown in FIG. **12**(*a*), FIG. **12**(*b*), and FIG. **12**(*c*) respectively. Data contained within an image element is encoded and must contain type information to indicate the specific image format.

The invention uses attribute based mechanisms for usage management. The policies defined over content must therefore consist of rules that address usage over an ontology of possible user attributes of concern. Of specific interest is a user's primary attributes: mission affiliation, clearance levels (both sensitivity and category), organization, and computational environment (consisting of both device and operating system).

FIG. **13** illustrates a table of all possible attributes for usage management decisions. Sets denote membership with no associated value. Orderings on the other hand have distinct values increasing from left to right in the listed enumerations. For example, a user can be affiliated with a specific mission in Domain A, either tropic_thunder or gallant_entry, or both. That user is also associated with a sensitivity value, either unclassified, secret, or top_secret, where top_secret is the most sensitive and unclassified the least. Need-to-use decisions are based on the current context in tandem with mission and organizational affiliation. Attribute based control is used in these scenarios, in which access decisions are made based on the attributes of a requesting user rather than defined roles.

FIG. **14** illustrates a table of all possible attributes for usage management decisions specific to users. User attributes support defined policy elements. Not every policy attribute has a corresponding user attribute as not all policy attributes are associated with users. Some are associated with the user's

environment, like operating system or device. Policies are evaluated either via direct set membership or via membership in a category in an ordering. Content can be affiliated with multiple sets with regard to set-oriented attributes. Likewise, users can belong to multiple sets as well. Both content and users may be associated with a single value from an ordering element, as that value dominates lower values as well. For example, a user can be affiliated with both the tropic_thunder and gallant_entry missions, but only one of the clearance values of uncleared, secret, or top secret. In the case of clearance values, secret subsumes uncleared, so a user with a secret attribute set would be able to access any unclassified material.

A domain specific language (DSL) is used to describe policies, for example, a Ruby-based DSL. In larger heterogeneous deployments, a standards-based alternative like an eXtensible Access Control Markup Language (XACML) may be more suitable. FIG. **15** illustrates a Domain Specific Language (DSL) file relating to policy domain specific language supporting a subset of XACML elements according to one embodiment of the invention. As shown in FIG. **15**, a base policy exists, p**1**, that all other policies inherit. That policy requires that all rules evaluate to true. Policy p**2** adds another rule based on devices, all of which must evaluate to true as well. Finally, policy p**3** adds two additional rules, only one of which must evaluate to true for the policy to be fulfilled.

Each of the defined components has an associated interface defined over domain datatypes. In one embodiment, these interfaces are implemented using Representational State Transfer (REST) semantics over Hypertext Transfer Protocol (HTTP), and the datatypes are represented in Extensible Markup Language (XML). As shown in FIG. **16**, the system primarily deals with two key datatypes, artifacts and policy_sets. For the purpose of networked data transfer, both of these datatypes are formatted strings of XML and policy DSL data. An artifact_descriptor combines an artifact with its associated set of policies. An artifact_descriptor_list is an unlimited sequence of artifact descriptors.

Network status information is contained in status elements and grouped into a context structure, as shown in FIG. **17**. A status_list is essentially a dictionary of network connection statuses organized by link name, where an edge is named by concatenating the edge nodes in any order. These node names are concatenated and separated by a pipe symbol, so that the edge between NodeA and NodeB is named NodeA|NodeB or NodeB|NodeA. This makes searching less efficient, in that a context manager component can contain a status_list with names in either ordering, in exchange for easier and more terse data exchange.

Finally, shown in FIG. **18**, the error exception is represented by standard HTTP error codes and responses operationally, and is used extensively throughout system interface operations. Other information can be included in exception messages if the errors are not HTTP specific.

The artifact_manager interface is shown in FIG. **19**. This interface is mapped to a REST style request over HTTP where the argument ordering is preserved when building the URL for accessing artifact content. For example, when accessing a specific artifact, the artifact operation called with a username of 'truchas', on a mobile device, for artifact X1234 would map to the URL http://host/artifact/truchas/mobiledevice/X1234. Likewise, a similar operation call on the artifacts operation would use the URL http://host/artifacts/truchas/mobiledevice. Both node components and router components implement the artifact_manager interface.

This type of calling convention is used throughout the system. The specific ordering of the URL elements stems from corresponding artifact set relationships. Specifically, the set of all artifacts a user has access to is the same as or larger than the set of all artifacts that a user on a specific device can access and is also the same size or smaller that the set of all available artifacts.

The context manager interface as shown in FIG. **20** describes how the network context monitor exposes network state information to requesters. Note, in this case, the defined interface maps to the URL http://host/context.

The usage_management_mechanism makes decisions with respect to proposed activities based on a set of policies and the current dynamic environmental context as shown in FIG. **21**.

The repository interface shown FIG. **22** defines how information is stored and retrieved within a given node or router. This is an internal component used for concrete data item storage.

Finally, the dispatcher interface shown in FIG. **23**, also used internally within a given node or router, describes how requests are passed to other network participants.

The system is implemented with primary interface and data type definitions. At this point, it can filter information through defined nodes implementing different strategies in accordance with specific defined rules. It is also instrumented so that it can generate accurate timing information needed to measure availability impacts of confidentiality strategies on transmitted information.

FIG. **24** illustrates an example of a cloud computing system **900** that may be used to implement the methods according to the invention. The cloud computing system **900** includes a plurality of interconnected computing environments. The cloud computing system **900** utilizes the resources from various networks as a collective virtual computer, where the services and applications can run independently from a particular computer or server configuration making hardware less important.

Specifically, the cloud computing system **900** includes at least one client computer **902**. The client computer **902** may be any device through the use of which a distributed computing environment may be accessed to perform the methods disclosed herein, for example, a traditional computer, portable computer, mobile phone, personal digital assistant, tablet to name a few. The client computer **902** includes memory such as random access memory (RAM), read-only memory (ROM), mass storage device, or any combination thereof. The memory functions as a computer usable storage medium, otherwise referred to as a computer readable storage medium, to store and/or access computer software and/or instructions.

The client computer **902** also includes a communications interface, for example, a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, wired or wireless systems, etc. The communications interface allows communication through transferred signals between the client computer **902** and external devices including networks such as the Internet **904** and cloud data center **906**. Communication may be implemented using wireless or wired capability such as cable, fiber optics, a phone line, a cellular phone link, radio waves or other communication channels.

The client computer **902** establishes communication with the Internet **904**—specifically to one or more servers—to, in turn, establish communication with one or more cloud data centers **906**. A cloud data center **906** includes one or more networks **910***a*, **910***b*, **910***c* managed through a cloud management system **908**. Each network **910***a*, **910***b*, **910***c* includes resource servers **912***a*, **912***b*, **912***c*, respectively. Servers **912***a*, **912***b*, **912***c* permit access to a collection of

computing resources and components that can be invoked to instantiate a virtual machine, process, or other resource for a limited or defined duration. For example, one group of resource servers can host and serve an operating system or components thereof to deliver and instantiate a virtual machine. Another group of resource servers can accept requests to host computing cycles or processor time, to supply a defined level of processing power for a virtual machine. A further group of resource servers can host and serve applications to load on an instantiation of a virtual machine, such as an email client, a browser application, a messaging application, or other applications or software.

The cloud management system **908** can comprise a dedicated or centralized server and/or other software, hardware, and network tools to communicate with one or more networks **910***a*, **910***b*, **910***c*, such as the Internet or other public or private network, with all sets of resource servers **912***a*, **912***b*, **912***c*. The cloud management system **908** may be configured to query and identify the computing resources and components managed by the set of resource servers **912***a*, **912***b*, **912***c* needed and available for use in the cloud data center **906**. Specifically, the cloud management system **908** may be configured to identify the hardware resources and components such as type and amount of processing power, type and amount of memory, type and amount of storage, type and amount of network bandwidth and the like, of the set of resource servers **912***a*, **912***b*, **912***c* needed and available for use in the cloud data center **906**. Likewise, the cloud management system **908** can be configured to identify the software resources and components, such as type of Operating System ("OS"), application programs, and the like, of the set of resource servers **912***a*, **912***b*, **912***c* needed and available for use in the cloud data center **906**.

The invention is also directed to computer products, otherwise referred to as computer program products, to provide software to the cloud computing system **900**. Computer products store software on any computer useable medium, known now or in the future. Such software, when executed, may implement the methods according to certain embodiments of the invention. Examples of computer useable mediums include, but are not limited to, primary storage devices (e.g., any type of random access memory), secondary storage devices (e.g., hard drives, floppy disks, CD ROMS, ZIP disks, tapes, magnetic storage devices, optical storage devices, Micro-Electro-Mechanical Systems (MEMS), nanotechnological storage device, etc.), and communication mediums (e.g., wired and wireless communications networks, local area networks, wide area networks, intranets, etc.). It is to be appreciated that the embodiments described herein may be implemented using software, hardware, firmware, or combinations thereof.

The cloud computing system **900** of FIG. **9** is provided only for purposes of illustration and does not limit the invention to this specific embodiment. It is appreciated that a person skilled in the relevant art knows how to program and implement the invention using any computer system or network architecture.

The described embodiments are to be considered in all respects only as illustrative and not restrictive, and the scope of the invention is not limited to the foregoing description. Those of skill in the art may recognize changes, substitutions, adaptations and other modifications that may nonetheless come within the scope of the invention and range of the invention.

The invention claimed is:

1. A network for transferring information across incompatible domains, comprising:

an application layer including a context manager client service component and a dispatch service component,

a domain layer including a context manager component and a usage management mechanism component,

one or more nodes, wherein the one or more nodes are dispersed in a security domain,

wherein a request for content is received by the domain layer, the context manager component of the domain layer determines an environmental context in which the content is to be used according to the request,

the usage management mechanism component of the domain layer associates one or more policies with the request based on the environmental context,

the dispatch service component of the application layer sends a query to the one or more nodes for the content based on the one or more policies associated to the request based on the environmental context, wherein the security domain is divided into a plurality of domains with each domain governed by a different policy,

further comprises the one or more nodes deliver content only when the policy of the domain adheres to the one or more policies associated to the request based on the environmental context; and

the one or more nodes removes a portion of content that does not adhere to the one or more policies associated to the request based on the environmental context before delivering the content.

2. The network for transferring information across incompatible domains according to claim **1** further comprising an infrastructure layer including an information and policy repository component containing content organized by the one or more policies associated to the request based on the environmental context.

3. The network for transferring information across incompatible domains according to claim **1** wherein the one or more nodes interface with an external data source comprising content.

4. The network for transferring information across incompatible domains according to claim **1** wherein the security domain is hierarchical.

5. The network for transferring information across incompatible domains according to claim **1** wherein the one or more nodes includes a router node.

6. The network for transferring information across incompatible domains according to claim **1** wherein the request for content uses a Hypertext Transfer Protocol (HTTP).

7. The network for transferring information across incompatible domains according to claim **1** wherein the one or more policies associated to the request based on the environmental context includes details how the content can be used.

* * * * *