

(51) International Patent Classification:
H04L 29/06 (2006.01)(21) International Application Number:
PCT/US2013/038557(22) International Filing Date:
29 April 2013 (29.04.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/461,069 1 May 2012 (01.05.2012) US(71) Applicant (for all designated States except US): **HARRIS CORPORATION** [US/US]; 1025 W. NASA Blvd., MS A-11I, Melbourne, Florida 32919 (US).(72) Inventors: **SMITH, Wayne B.**; 507 Harland Avenue, Melbourne Beach, Florida 32951 (US). **LIN, Ellen K.**; 2460 Brookshire Circle, West Melbourne, Florida 32904 (US).(74) Agents: **YATSKO, Michael S.** et al.; Harris Corporation, 1025 W. NASA Blvd., MS A-11I, Melbourne, Florida 32919 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))

(54) Title: SYSTEMS AND METHODS FOR IDENTIFYING, DETERRING AND/OR DELAYING ATTACKS TO A NETWORK USING SHADOW NETWORKING TECHNIQUES

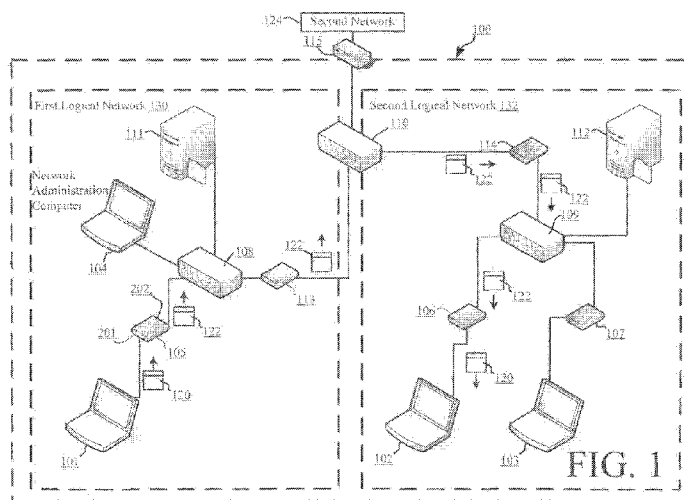


FIG. 1

(57) Abstract: Systems (100) and methods (2100) for identifying, deterring and/or delaying malicious attacks being waged on a Computer Network ("CN"). The methods involve implementing a Mission Plan ("MP") at a first Network Node ("NN"). MP (1900, 1902) specifies that: a first IDentity Parameter ("IDP") for a second NN has numerous possible values associated therewith; and at least two possible values are to be used in communications to and from the second NN in different timeslots of a time frame (2020-2026). At the first NN, a value for the first IDP, which is contained in a received packet, is compared with the possible values specified in MP to determine if the value is a "correct" value for a current timeslot. If it is determined that the value is not "correct" for the current timeslot, then the first NN performs actions to identify, deter or delay a possible malicious attack on CN.

SYSTEMS AND METHODS FOR IDENTIFYING, DETERRING AND/OR DELAYING ATTACKS TO A NETWORK USING SHADOW NETWORKING TECHNIQUES

5 BACKGROUND OF THE INVENTION

Statement of the Technical Field

The inventive arrangements relate to computer network security, and more particularly to systems for communicating between two or more logical subdivisions of a computer network where the network is dynamically maneuverable to defend against malicious attacks.

Description of the Related Art

The central weakness of current cyber infrastructure is its static nature. Assets receive permanent or infrequently-changing identifications, allowing adversaries nearly unlimited time to probe networks, map and exploit vulnerabilities. Additionally, data traveling between these fixed entities can be captured and attributed. The current approach to cyber security places technologies such as firewalls and intrusion detection systems around fixed assets, and uses encryption to protect data en route. However, this traditional approach is fundamentally flawed because it provides a fixed target for attackers. In today's globally connected communications infrastructure, static networks are vulnerable networks.

The Defense Advanced Research Projects Agency (“DARPA”) Information Assurance (“IA”) Program has performed initial research in the area of dynamic network defense. A technique was developed under the IA program to dynamically reassign Internet protocol (“IP”) address space feeding into a pre-designated network enclave for the purpose of confusing any would-be adversaries observing the network. This technique is called DYNamic Network Address Transformation (“DYNAT”). An overview of the DYNAT technology was presented in a paper by DARPA entitled “Dynamic Approaches to Thwart Adversary Intelligence” which was published in 2001.

SUMMARY OF THE INVENTION

Embodiments of the present invention concern systems and methods for identifying, deterring and/or delaying malicious attacks being waged on a computer network. The methods involve implementing a mission plan at a first network node.

- 5 The mission plan specifies that: at least one first identity parameter for a second network node has a plurality of possible values associated therewith; and at least two of the possible values are to be used in communications to and from the second network node in different timeslots of a plurality of timeslots of a time frame. At the first network node, a value for the first identity parameter, which is contained in a
- 10 received packet, is compared with the possible values specified in the mission plan to determine if the value is a “correct” value for a current timeslot (i.e., one of the possible values which is to be used in communications to and from the second network node in the current timeslot, as specified by the mission plan). If it is determined that the value is not “correct” for the current timeslot, then the first
- 15 network node performs at least one action to identify, deter or delay a possible malicious attack on the computer network. The action includes, but is not limited to, dropping the received packet, forwarding the received packet to an intrusion analysis tool or a honeypot analysis tool, logging information associated with the received packet, determining a source of the received packet, or sending a response to the
- 20 source of the received packet.

BRIEF DESCRIPTION OF THE DRAWINGS

- Embodiments will be described with reference to the following drawing figures, in which like numerals represent like items throughout the figures, and in
- 25 which:

FIG. 1 is an example of a computer network that is useful for understanding the present invention.

FIG. 2 is an example of a module that can be used in the present invention for performing certain manipulations of identity parameters.

FIG. 3 is a drawing that is useful for understanding a tool that can be used to help characterize the network in FIG. 1.

FIG. 4 is an example of a dialog box of a Graphical User Interface (“GUI”) that can be used to select dynamic settings for modules in FIG. 1.

5 FIG. 5 is an example of a dialog box of a GUI that can be used to select a sequence of active states and bypass states associated with each module in FIG. 1.

FIG. 6 is a diagram that is useful for understanding the way in which a mission plan can be communicated to a plurality of modules in the network in FIG. 1.

FIG. 7 is an example of a dialog box of a GUI that can be used to select a
10 mission plan and communicate the mission plan to the modules as shown in FIG. 6.

FIG. 8 is a flowchart that is useful for understanding the operation of a module in FIG. 1.

FIG. 9 is a flowchart that is useful for understanding the operation of a Network Control Software Application (“NCSA”) in relation to creating and loading
15 mission plans.

FIG. 10 is a block diagram of a computer architecture that can be used to implement the modules in FIG. 1.

FIG. 11 is a schematic illustration of a conventional protocol stack.

FIG. 12 is a schematic illustration of a conventional packet.

20 FIG. 13 is a schematic illustration of a Moving Target Technology (“MTT”) protocol stack.

FIG. 14 is a schematic illustration of an MTT packet.

FIG. 15 is a schematic illustration that is useful for understanding the operations of a module configured to translate identity parameters.

25 FIGS. 16-17 each provide a flow diagram of an exemplary process for changing at least one identity parameter of a packet.

FIG. 18 is a block diagram of a computer architecture that can be used to implement a Network Administration Computer (“NAC”) shown in FIG. 1.

FIG. 19 is a schematic illustration of exemplary mission plans.

FIGS. 20A-20E each provide an exemplary time structure for at least one mission plan shown in FIG. 19.

FIG. 21 is a flow diagram of an exemplary method for shadow networking.

5

DETAILED DESCRIPTION

The invention is described with reference to the attached figures. The figures are not drawn to scale and they are provided merely to illustrate the instant invention. Several aspects of the invention are described below with reference to example applications for illustration. It should be understood that numerous specific
10 details, relationships, and methods are set forth to provide a full understanding of the invention. One having ordinary skill in the relevant art, however, will readily recognize that the invention can be practiced without one or more of the specific details or with other methods. In other instances, well-known structures or operations are not shown in detail to avoid obscuring the invention. The invention is not limited
15 by the illustrated ordering of acts or events, as some acts may occur in different orders and/or concurrently with other acts or events. Furthermore, not all illustrated acts or events are required to implement a methodology in accordance with the invention.

It should also be appreciated that the terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting
20 of the invention. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. Furthermore, to the extent that the terms "including", "includes", "having", "has", "with", or variants thereof are used in either the detailed description and/or the claims, such terms are intended to be inclusive in a manner similar to the term "comprising."

25 Further, unless otherwise defined, all terms (including technical and scientific terms) used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this invention belongs. It will be further understood that terms, such as those defined in commonly used dictionaries, should be interpreted as having a meaning that is consistent with their meaning in the context of

the relevant art and will not be interpreted in an idealized or overly formal sense unless expressly so defined herein.

Identity Agile Computer Network

Referring now to FIG. 1, there is shown a diagram of an exemplary
5 computer network **100** which includes a plurality of computing devices. The
computing devices can include client computers **101-103**, NAC **104**, servers **111, 112**,
network layer 2 switches **108, 109**, layer 3 switch **110**, and a bridge **115**. The client
computers **101-103** can be any type of computing device which might require network
services, such as a conventional tablet, notebook, laptop or desktop computer. The
10 layer 3 switch **110** can be a conventional routing device that routes data packets
between computer networks. The layer 2 switches **108, 109** are conventional hub
devices (e.g., an Ethernet hub) as are well known in the art. Servers **111, 112** can
provide various computing services utilized by client computers **101-103**. For
example, the servers **111, 112** can be file servers which provide a location for shared
15 storage of computer files used by client computers **101-103**.

The communication media for the computer network **100** can be wired,
wireless or both, but shall be described herein as a wired network for simplicity and to
avoid obscuring the invention. The network will communicate data using a
communication protocol. As is well known in the art, the communication protocol
20 defines the formats and rules used for communicating data throughout the network.
The computer network **100** in FIG. 1 can use any communication protocol or
combination of protocols which is now known or known in the future. For example,
the computer network **100** can use the well known Ethernet protocol suite for such
communications. Alternatively, the computer network **100** can make use of other
25 protocols, such as the protocols of an internet protocol suite (often referred to as the
TCP/IP suite), Synchronous Optical NETWORK/Synchronous Digital Hierarchy
("SONET/SDH") based protocols, or Asynchronous Transfer Mode ("ATM")
communication protocols. In some embodiments, one or more of these
communication protocols can be used in combination. Although one network
30 topology is shown in FIG. 1, the invention is not limited in this regard. Instead, any

type of suitable network topology can be used, such as a bus network, a star network, a ring network or a mesh network.

The invention generally concerns a method for communicating data in a computer network (e.g., in computer network **100**), where data is communicated from a first computing device to a second computing device. Computing devices within the network are represented with multiple identity parameters. The phrase “identity parameters”, as used herein, can include items such as an IP address, a Media Access Control (“MAC”) address, a port number and so on. However, the invention is not limited in this regard, and the identity parameters can also include a variety of other information which is useful for characterizing a network node. The various types of identity parameters contemplated herein are discussed below in further detail.

The inventive arrangements involve the use of MTT to manipulate one or more of such identity parameters for one or more computing devices within the computer network **100**. This technique disguises communication patterns and network addresses of such computing devices. The manipulation of identity parameters as described herein is generally performed in conjunction with data communications in the computer network **100**, i.e., when data is to be communicated from a first computer in the network (e.g., client computer **101**) to a second computer in the network (e.g., client computer **102**). Accordingly, the identity parameters that are manipulated can include those of a source computing device (i.e., the device from which the data originated) and the destination computing device (i.e., the device to which the data is being sent). The set of identity parameters that is communicated is referred to herein as an IDentity Parameter (“IDP”) set. This concept is illustrated in FIG. 1, which shows that an IDP set **120** is transmitted by client computer **101** as part of a data packet (not shown).

The process according to the inventive arrangements involves selectively modifying at a first location within the computer network **100**, values contained in a data packet or datagram which specify one or more identify parameters of a source computing device and/or a destination computing device. The identity parameters are modified in accordance with a mission plan. The location where such modification is

performed will generally coincide with the location of one module **105-107, 113, 114** of the computer network **100**. Referring once again to FIG. 1, it can be observed that the modules **105-107, 113, 114** are interposed in the computer network **100** between the various computing devices which comprise nodes in such network. In these
5 locations, the modules **105-107, 113, 114** intercept data packet communications, perform the necessary manipulations of identity parameters, and retransmit the data packets along a transmission path. In alternative embodiments, the modules **105-107, 113, 114** can perform a similar function, but can be integrated directly into one or more of the computing devices. For example, the modules could be integrated into
10 client computers **101, 102, 103**, servers **111, 112**, layer 2 switches **108, 109** and/or layer 3 switch **110**. In this scenario, the modules can comprise hardware that is added to the computing and/or software that is installed on the computing device **101-103, 108-112**. In some software embodiments, the modules are implemented as kernel mode software (e.g., as device drivers) that modifies the identity parameters.

15 Additionally, the computer network **100** can be divided into a number of logical subdivisions, sometimes referred to as sub-networks or subnets, connected through layer 3 switch **110**. An enterprise network can be divided into a number of subnets for a variety of administrative or technical reasons including, but not limited to, hiding the topology of the network from being visible to external hosts, connecting
20 networks utilizing different network protocols, separately administering network addressing schemes on the subnet level, enabling management of data traffic across subnets due to constrained data connections, and the like. Subnetting is well known in the art and will not be described in further detail.

Referring again to FIG. 1, the computer network **100** is divided into two
25 logical networks, namely a first logical network **130** and a second logical network **132**. The phrase “logical network”, as used herein, refers to any logical subdivision of a computer network. In an embodiment, logical networks **130, 132** are connected through layer 3 switch **110**. Layer 3 switch **110** is responsible for directing traffic between the logical networks (i.e., from client computer **101** to client computer **103**).
30 Layer 3 switch **110** is also responsible for directing traffic from any host connected to

the computer network **100** bound for a second network **124**. In the embodiment shown in FIG. 1, traffic routed from the computer network **100** to the second network **124** passes through bridge **115**. As with the modules above, the functionality of the bridge **115** could be integrated within the layer 3 switch **110**.

5 An example of a functional block diagram of a module **105** is shown in FIG. 2. Modules **106**, **107**, **113**, **114** of FIG. 1 can have a similar functional block diagram as that shown in FIG. 2, but it should be understood that the invention is not limited in this regard. As shown in FIG. 2, the module **105** has at least two data ports **201**, **202**, each of which can correspond to a respective network interface device **204**,
10 **205**. Data received at data port **201** is processed at network interface device **204** and temporarily stored at an input buffer **210**. The processor **215** accesses the input data packets contained in input buffer **210** and performs any necessary manipulation of identity parameters as described herein. The modified data packets are passed to output buffer **212** and subsequently transmitted from data port **202** using network
15 interface device **205**. Similarly, data received at data port **202** is processed at network interface device **205** and temporarily stored at an input buffer **208**. The processor **215** accesses the input data packets contained in input buffer **208** and performs any necessary manipulation of identity parameters as described herein. The modified data packets are passed to an output buffer **206** and subsequently transmitted from data
20 port **201** using network interface device **204**. In module **105**, manipulation of identity parameters is performed by processor **215** in accordance with a mission plan **220** stored in a memory **218**.

 It will be understood from FIG. 2 that the module **105** is preferably configured so that it operates bi-directionally. In such embodiments, the module **105**
25 can implement different modification functions, depending on a source of a particular data packet. The dynamic modification function in the module **105** can be specified in the mission plan in accordance with a source computing device of a particular data packet. The module **105** can determine a source of data packets by any suitable means. For example, a source address of a data packet can be used for this purpose.

During operation, the processor **215** will determine one or more false identity parameter values that are to be used in place of the true identity parameter values. The processor **215** will transform one or more true identity parameter values to one or more false identity parameter values which are preferably specified by a
5 pseudorandom function. Following this transformation, the module **105** will forward the modified packet or datagram to the next node of the computer network **100** along a transmission path. At subsequent points in the communication path, an adversary who is monitoring such network communications will observe false or incorrect information about the identity of computing devices communicating on the computer
10 network **100**.

In a preferred embodiment, the false identity parameters that are specified by the pseudorandom function are varied in accordance with the occurrence of at least one proactive trigger event or at least one reactive trigger event. A proactive/reactive trigger event causes the processor **215** to use the pseudorandom function to generate a
15 new set of false identity parameter values into which the true identity parameters are transformed. Accordingly, the proactive/reactive trigger event serves as a basis for the dynamic variation of the false identity parameters described herein. Proactive and reactive trigger events are discussed in more detail below. However, it should be noted that proactive/reactive trigger events for selecting a new set of false values for
20 identity parameters can be based on at least one pre-defined rule. The rule comprises a statement that defines at least one proactive or reactive trigger event. In this regard, the user rule may implement a proactive triggering scheme or a reactive triggering scheme. A proactive triggering scheme comprises a time based scheme. A reactive triggering scheme comprises a user activation based scheme, a packet inspection
25 based scheme, congestion level based scheme, a heuristic algorithm based scheme and/or a Network-Based Attack (“NBA”) analysis based scheme. Each of the listed schemes will be described in detail below.

The transformation of identity parameters described above provides one way to maneuver a computer network **100** for purposes of thwarting a cyber attack.
30 In a preferred embodiment, the mission plan **220** implemented by processor **215** will

also control certain other aspects of the manner in which computer network **100** can maneuver. For example, the mission plan **220** can specify that a dynamic selection of identity parameters is manipulated. The dynamic selection can include a choice of which identity parameters are selected for modification, and/or a number of such

5 identity parameters that are selected. This variable selection process provides an added dimension of uncertainty or variation which can be used to further thwart an adversary's effort to infiltrate or learn about a computer network **100**. As an example of this technique, consider that during a first time period, the module **105** can modify a destination IP address and a destination MAC address of each data packet. During a

10 second time period, the module **105** could manipulate the source IP address and a source host name in each data packet. During a third period of time, the module **105** could manipulate a source port number and a source user name. Changes in the selection of identity parameters can occur synchronously (i.e., all selected identity parameters are changed at the same time). Alternatively, changes in the selection of

15 identity parameters can occur asynchronously (i.e., the group of selected identity parameters changes incrementally as individual identity parameters are added or removed from the group of selected identity parameters).

A pseudorandom function is preferably used for determining the selection of identity values that are to be manipulated or transformed into false values. In other

20 words, the module **105** will transform only the identity parameters selected by the pseudo-random function. In a preferred embodiment, the selection of identity parameters that are specified by the pseudorandom function is varied in accordance with the occurrence of a proactive/reactive trigger event. The proactive/reactive trigger event causes processor **215** to use a pseudorandom function to generate a new

25 selection of identity parameters which are to be transformed into false identity parameters. Accordingly, the proactive/reactive trigger event serves as a basis for the dynamic variation of the selection of identity parameters described herein. Notably, the values of the identity parameters can also be varied in accordance with a pseudorandom algorithm.

The module **105** is advantageously capable of also providing a third method of maneuvering the computer network for purposes of thwarting a cyber attack. Specifically, the mission plan **220** loaded in module **105** can dynamically vary the location within the network where the modification or transformation of the identity parameters takes place. Consider that modification of identity parameters in an IDP set **120** sent from client computer **101** to client computer **102** could occur in module **105**. This condition is shown in FIG. 1, where the identity parameters contained in IDP set **120** are manipulated in module **105** so that the IDP set **120** is transformed to a new or modified IDP set **122**. At least some of the identity parameters in the IDP set **122** are different as compared to the identity parameters in the IDP set **120**. But, the location where such transformation occurs is preferably also controlled by the mission plan. Accordingly, manipulation of the IDP set **120** could, for example, sometimes occur at module **113** or **114** of FIG. 1, instead of at module **105**. This ability to selectively vary the location where manipulation of identity parameters occurs adds a further important dimension to the maneuvering capability of the computer network **100**.

The dynamic variation in the location where identity parameters are modified is facilitated by selectively controlling an operating state of each module **105-107, 113, 114** of FIG. 1. To that end, the operational states of each module **105-107, 113, 114** of FIG. 1 preferably includes (1) an active state in which data is processed in accordance with a current mission plan, and (2) a by-pass state in which packets can flow through the module as if the module was not present. The location where the dynamic modification is performed is controlled by selectively causing certain modules of the computer network **100** to be in an active state and certain modules of the computer network **100** to be in a standby state. The location can be dynamically changed by dynamically varying the current state of the modules **105-107, 113, 114** of FIG. 1 in a coordinated manner.

The mission plan **220** can include a predefined sequence for determining the locations within the computer network **100** where the identity parameters are to be manipulated. Locations where identity parameters are to be manipulated will change

in accordance with the predefined sequence at times indicated by a proactive/reactive trigger event. For example, the proactive/reactive trigger event can cause a transition to a new location for manipulation or transformation of identity parameters as described herein. Accordingly, the proactive/reactive trigger event serves as a basis
5 for the occurrence of a change in the location where identity parameters are modified, and the predefined sequence determines where the new location will be.

From the foregoing, it will be appreciated that a data packet is modified at a module **105-107, 113, 114** of FIG. 1 to include false identity parameters. At some point within the computer network **100**, it is necessary to restore the identity
10 parameters to their true values, so that the identity parameters can be used to properly perform their intended function in accordance with the particular network protocol. Accordingly, the inventive arrangements also includes dynamically modifying, at a second location (i.e., a second module), the assigned values for the identity parameters in accordance with the mission plan **220**. The modification at the second
15 location essentially comprises an inverse of a process used at the first location to modify the identity parameters. The module at the second location can thus restore or transform the false value identity parameters back to their true values. In order to accomplish this action, the module at the second location must be able to determine at least (1) a selection of identity parameter value that are to be transformed, and (2) a
20 correct transformation of the selected identity parameters from false values to true values. In effect, this process involves an inverse of the pseudorandom process or processes used to determine the identity parameter selection and the changes effected to such identity parameter values. The inverse transformation step is illustrated in FIG. 1, where the IDP set **122** is received at module **106**, and the identity parameter
25 values in IDP set **122** are transformed or manipulated back to their original or true values. In this scenario, module **106** converts the identity parameters values back to those of IDP set **120**.

Notably, a module must have some way of determining the proper transformation or manipulation to apply to each data communication it receives. In a
30 preferred embodiment, this determination is performed by examining at least a source

address identity parameter contained within the received data communication. For example, the source address identity parameter can include an IP address of a source computing device. Once the true identity of the source computing device is known, the module consults the mission plan (or information derived from the mission plan) to determine what actions it needs to take. For example, these actions could include converting certain true identity parameter values to false identity parameter values. Alternatively, these changes could include converting false identity parameter values back to true identity parameter values.

Notably, there will be instances where the source address identity parameter information contained in a received data communication has been changed to a false value. In those circumstances, the module receiving the data communication will not immediately be able to determine the identity of the source of the data communication. However, the module which received the communication can in such instances still identify the source computing device. This is accomplished at the receiving module by comparing the false source address identity parameter value to a Look-Up-Table ("LUT") which lists all such false source address identity parameter values in use during a particular time. The LUT also includes a list of true source address identity parameter values that correspond to the false source address values. The LUT can be provided directly by the mission plan **220** or can be generated by information contained within the mission plan **220**. In either case, the identification of a true source address identity parameter value can be easily determined from the LUT. Once the true source address identity parameter has been determined, then the module which received the data communication can use this information to determine (based on the mission plan) what manipulations to the identity parameters are needed.

Notably, the mission plan **220** can also specify a variation in the second location where identity parameters are restored to their true values. For example, assume that the identity parameters are dynamically modified at a first location comprising module **105**. The mission plan can specify that the restoration of the identity parameters to their true values occurs at module **106** as described, but can

alternatively specify that dynamic modification occur instead at module **113** or **114**. In some embodiments, the location where such manipulations occur is dynamically determined by the mission plan in accordance with a predefined sequence. The predefined sequence can determine the sequence of locations or modules where the manipulation of identity parameters will occur.

The transition involving dynamic modification at different locations preferably occurs in accordance with a proactive/reactive trigger event. Accordingly, the predefined sequence determines the pattern or sequence of locations where data manipulations will occur, and the proactive/reactive trigger event serves as a basis for causing the transition from one location to the next. Proactive/reactive trigger events are discussed in more detail below; however, it should be noted that proactive/reactive trigger events can be based on at least one pre-defined rule. The rule comprises a statement that defines at least one proactive/reactive trigger event. In this regard, the rule may implement proactive triggering schemes or reactive triggering schemes. A proactive triggering scheme comprises a time based scheme. A reactive triggering scheme comprises a user activation based scheme, a packet inspection based scheme, a congestion level based scheme, a heuristic algorithm based scheme and/or an NBA analysis based scheme. Each of the listed schemes will be described below in detail. Control over the choice of a second location (i.e., where identity parameters are returned to their true values) can be effected in the same manner as described above with regard to the first location. Specifically, operating states of two or more modules can be toggled between an active state and a bypass state. Manipulation of identity parameters will only occur in the module which has an active operating state. The module with a bypass operating state will simply pass data packets without modification.

Alternative methods can also be used for controlling the location where manipulation of identity parameters will occur. For example, a network administrator can define in a mission plan several possible modules where identity parameters can be converted from true values to false values. Upon the occurrence of a proactive/reactive trigger event, a new location can be selected from among the

several modules by using a pseudorandom function, and using a trigger time as a seed value for the pseudorandom function. If each module implements the same pseudorandom function using the same initial seed values then each module will calculate the same pseudorandom value. The trigger time can be determined based on a clock time (such as a GPS time or system clock time). In this way, each module can independently determine whether it is currently an active location where manipulation of identity parameters should occur. Similarly, the network administrator can define in a mission plan several possible modules where dynamic manipulation returns the identity parameters to their true values. The selection of which module is used for this purpose can also be determined in accordance with a trigger time and a pseudorandom function as described herein. Other methods are also possible for determining the location or module where identity parameter manipulations are to occur. Accordingly, the invention is not intended to be limited to the particular methods described herein.

Notably, varying the position of the first and/or second locations where identity functions are manipulated will often result in varying a physical distance between the first and second location along a network communication path. The distance between the first and second locations is referred to herein as a distance vector. The distance vector can be an actual physical distance along a communication path between the first and second location. However, it is useful to think of the distance vector as representing the number of network nodes that are present in a communication path between the first and second locations. It will be appreciated that dynamically choosing different positions for the first and second locations within the network can have the effect of changing the number of nodes between the first and second locations. For example, in FIG. 1, the dynamic modification of identity parameters are implemented in selected ones of the modules **105, 106, 107, 113, 114**. The modules actually used to respectively implement the dynamic modification is determined as previously described. If module **105** is used for converting identity parameters to false values and module **106** is used to convert them back to true values, then there are three network nodes (**108, 110, 109**) between modules **105** and **106**.

But if module **113** is used to convert to false values and module **114** is used to convert the identity parameters back to true values, then there is only one network node **110** between modules **113** and **114**. Accordingly, it will be appreciated that dynamically changing the position of locations where dynamic modification occurs can

5 dynamically vary the distance vector. This variation of the distance vector provides an added dimension of variability to network maneuvering or modification as described herein.

In the present invention, the manipulation of identity parameter values, the selection of identity parameters, and the locations where these identity parameters is

10 each defined as a maneuvering parameter. Whenever a change occurs in one of these three maneuvering parameters, it can be said that a network maneuver has occurred. Any time one of these three maneuvering parameters is changed, we can say that a network maneuver has occurred. In order to most effectively thwart an adversary's efforts to infiltrate a computer network **100**, network maneuvering is preferably

15 controlled by means of a pseudorandom process as previously described. Those skilled in the art will appreciate that a chaotic process can also be used for performing this function. Chaotic processes are technically different as compared to pseudorandom functions, but for purposes of the present invention, either can be used, and the two are considered equivalent. In some embodiments, the same

20 pseudorandom process can be used for dynamically varying two or more of the maneuvering parameters. However, in a preferred embodiment of the invention, two or more different pseudorandom processes are used so that two or more of these maneuvering parameters are modified independently of the others.

Proactive and Reactive Trigger Events

25 As noted above, the dynamic changes to each of the maneuvering parameters is controlled by at least one proactive trigger or reactive trigger. A proactive trigger is a pre-defined event that causes a change to occur in relation to the dynamic modifications described herein. In contrast, a reactive trigger is a purely spontaneous or user initiated event that causes a change to occur in relation to the

30 dynamic modifications described herein. Stated differently, it can be said that the

proactive or reactive trigger causes the network to maneuver in a new way that is different than at a previous time (i.e., before the occurrence of the proactive or reactive trigger). For example, during a first period of time, a mission plan or security model can cause an IP address to be changed from value A to value B; but after the
5 proactive/reactive trigger event, the IP address can instead be changed from value A to value C. Similarly, during a first period of time a mission plan or security model can cause an IP address and a MAC address to be modified; but after the proactive/reactive trigger event, the mission plan or security model can instead cause a MAC address and a user name to be modified.

10 In its simplest form a proactive trigger event can be based on a time based scheme. In a time based scheme, a clock time in each module could serve as a trigger. For example, a trigger event could be defined as occurring at the expiration of every N (e.g., sixty) second time interval. For such an arrangement, one or more of the maneuvering parameters could change every N seconds in accordance with a
15 predetermined clock time. In some embodiments, all of the maneuvering parameters can change concurrently so that the changes are synchronized. In a slightly more complex embodiment, a time-based trigger arrangement can also be used, but a different unique trigger time interval can be selected for each maneuvering parameter. Thus, false identity parameter values could be changed at time interval X, a selection
20 of identity parameters would change in accordance with a time interval Y, and a location where such changes are performed would occur at time interval Z, where X, Y and Z are different values.

It will be appreciated that in embodiments of the invention which rely upon clock time as a trigger mechanism, it is advantageous to provide synchronization
25 as between the clocks in various modules **105, 106, 107, 113, 114** to ensure that packets are not lost or dropped due to unrecognized identity parameters. Synchronization methods are well known and any suitable synchronization mechanism can be used for this purpose. For example, the modules could be synchronized by using a highly accurate time reference such as a GPS clock time.

Alternatively, a unique wireless synchronization signal could be broadcast to each of the modules from a central control facility.

In its simplest form a reactive trigger can be based on a user activation based scheme, a packet inspection based scheme, a congestion level based scheme, a heuristic algorithm based scheme and/or an NBA analysis based scheme. In the user
5 activation based scheme, a user-software interaction defines a trigger event. For example, a trigger event occurs when a user of a computing device (e.g., computing device **101-103** of FIG. 1) depresses a given button of a user interface.

The packet inspection based scheme can involve analyzing a packet to
10 obtain an identifier identifying an origin of the packet, a destination of the packet, a group to which the origin or destination belong, and/or a type of payload contained in the packet. The packet inspection based scheme can also involve analyzing the packet to determine whether a code word is contained therein. Techniques for achieving such a packet inspection are well known in the art. Any such technique that is now
15 known or known in the future can be used with the present invention without limitation. In some embodiments, a reactive trigger event occurs when a value of the identifier matches a predefined value.

In the packet inspection scenarios, the inclusion of a particular type of content in a packet serves as a trigger or as a parameter for selecting a timing scheme
20 on which a trigger is based. For example, a trigger event could be defined as occurring (a) when a particular person of an entity (e.g., a commander of a military unit) communicates information to other members of the entity, and/or (b) when a particular code word is contained within the packet. Alternatively or additionally, a trigger event could be defined as occurring at the expiration of every N second time
25 interval as defined by a timing scheme selected in accordance with a particular packet inspection application, where N is an integer. In this regard, it should be understood that in some embodiments a first timing scheme can be selected (a) when a first person of an entity (e.g., a commander of a military unit) requests a communication session with other members of the entity or (b) when a particular code word exists
30 within a packet. A second different timing scheme can be selected (a) when a second

person of an entity (e.g., a lieutenant commander of a military unit) requests a communication session with other members of the entity or (b) when a second code word exists within a packet, and so on. Embodiments of the present invention are not limited to the particularities of the above provided examples. In this regard, it should
5 be understood that other content included in a packet can define a trigger event. For example, if the payload of a packet includes sensitive or confidential information, then a new mission plan or security model can be selected in accordance with the level of sensitivity or confidentiality of said information.

For such time-based trigger arrangements, one or more of the maneuvering
10 parameters could change every N (e.g., 60) seconds in accordance with a predetermined clock time. In some embodiments, all of the maneuvering parameters can change concurrently so that the changes are synchronized. In a slightly more complex embodiment, a time-based trigger arrangement can also be used, but a different unique trigger time interval can be selected for each maneuvering parameter.
15 Thus, false identity parameter values could be changed at time interval X, a selection of identity parameters would change in accordance with a time interval Y, and a location where such changes are performed would occur at time interval Z, where X, Y and Z are different values.

The congestion level based scheme can involve: monitoring and tracking
20 the level of congestion within a computer network; comparing a current level of congestion with a threshold value; and selecting a mission plan or security model from a plurality of mission plans/models based on the results of the comparison. In some scenarios, a new mission plan or security model is selected when the current level of congestion is equal to, greater than or less than the threshold value. In this
25 way, a mission plan or security model change occurs at apparently erratic time intervals based on changes in the level of congestion within a computer network.

The heuristic algorithm based scheme can involve analyzing a network to determine a state thereof. Such a network analysis can involve monitoring traffic patterns (e.g., the number of users), protocol patterns, and/or entropy patterns (i.e.,
30 who is communicating with who) of a network at particular times of a day. A traffic

pattern can be determined by collecting information about network equipment usage (e.g., a processor's usage) and a number of connections that exist from a network device (e.g., a network server). The collected information can be compared against the contents of a pre-defined table or matrix to identify which of a plurality of possible traffic patterns currently exists within a computer network. Based at least on the results of this comparison operation, a new mission plan or security model can be selected from a plurality of mission plans and/or security models for utilization in the computer network.

In some heuristic scenarios, the mission plans and/or security models can be configured such that a constant high level of traffic is maintained within a computer network despite changes in the amount of actual traffic therein. The constant high level of traffic is maintained by adjusting (i.e., increasing or decreasing) a noise level of a network in accordance with the amount of actual traffic therein. Consequently, the amount of actual traffic and the type of traffic pattern at any given time is masked.

A protocol pattern can be determined by collecting information about user activities related to network resources. Such information can include, but is not limited to, a history of user activities for at least one user of a computer network, times that user activities start, times that user activities stop, times that user activities have elapsed, and information identifying concurrent user activities being performed by at least one user of a computer network. The collected information can be analyzed to determine if a certain protocol pattern currently exists. If it is determined that a particular protocol pattern currently exists, then a new mission plan or security model can be selected from a plurality of mission plans/models for utilization in the computer network. In this way, a mission plan or security model change occurs at apparently erratic time intervals based on changes in protocol patterns (more particularly, changes in user activities).

The entropy pattern can be determined by collecting information about who is communicating with each other over the computer network. Based on the collected information, a new mission plan or security model is selected from a

plurality of mission plans/models for utilization in the computer network. In this scenario, a mission plan or security model change occurs at apparently erratic time intervals based on changes of the parties participating in communication sessions.

The NBA analysis is performed for purposes of determining a potential security threat, the level of an NBA, a type of an NBA, and/or the number of NBA attacks currently being waged on a computer network. Such NBA analyses are well known in the art, and therefore will not be described herein. Still, it should be understood that such NBA analyses can involve: monitoring and tracking attack events within a computer network; and performing LUT operations for purposes of: determining if there is a potential security threat; and/or determining the level of an NBA attack and/or the type of an NBA attack. Any NBA analysis technique that is now known or known in the future can be used with the present invention without limitation. Once the NBA analysis is completed, a new mission plan or security model can be selected from a plurality of mission plans/models for utilization in the computer network based on the results of the NBA analysis. For example, if it has been determined that an NBA is a low level NBA and/or is of a first type, then a first mission plan or security model is selected from a plurality of mission plans or security models. In contrast, if it has been determined that the NBA is a high level NBA and/or is of a second type, then a second different mission plan or security model is selected from the plurality of mission plans or security models. In this scenario, a mission plan or security model change occurs at apparently erratic time intervals based on changes in the level of NBA attacks and/or the types of NBA attacks. Additionally or alternatively, a new mission plan or security model can be selected when two or more NBA attacks of the same or different levels and/or types are currently being waged on the computer network. In this scenario, a mission plan or security model change occurs at apparently erratic time intervals based on changes in the number of attacks currently being performed.

In embodiments of the present invention, an NBA can be identified by a network security software suite. Alternatively, the NBA can be identified upon the receipt of a data packet at a module **105, 106, 107, 113, 114** where the packet contains

one or more identity parameters that are inconsistent with the present state of network maneuvering. Regardless of the basis for identifying an NBA, the existence of such NBA can serve as a reactive trigger event as described above.

Proactive/reactive trigger events based on the above described schemes
5 can cause the same types of network maneuvers. For example, false identity parameters, the selection of identity parameters and the locations of identity parameter transformations could remain stable (i.e., unchanged) except in the case where one or more of the following is detected: a clock time; a packet having a particular origin or destination; a code word contained in a packet; secret or confidential information
10 contained in a packet; a particular level of congestion; a particular traffic pattern; a particular protocol pattern; a particular entropy pattern; a security threat; an NBA of a particular level and/or type; and a particular number of NBAs currently being waged on a computer network. Such an arrangement might be chosen, for example, in computer networks where frequent network maneuvering is desirable so as to increase
15 the security thereof.

Alternatively, proactive/reactive trigger events based on the above described schemes can cause different types of network maneuvers. In such embodiments, a trigger event based on the results of an NBA analysis can have a different effect on the network maneuvering as compared to a trigger event based on
20 the results of a packet inspection and/or a heuristic algorithm. For example, an NBA-based trigger event can cause strategic or defensive changes in the network maneuvering so as to more aggressively counter such NBAs. The precise nature of such measures can depend on the nature of the threat, but can include a variety of responses. For example, different pseudorandom algorithms can be selected, and/or
25 the number of identity parameters selected for manipulation in each IDP set **120** can be increased. Also, the response can include increasing a frequency of network maneuvering. Thus, more frequent changes can be made with respect to (1) the false identity parameter values, (2) the selection of identity parameters to be changed in each IDP set, and/or (3) the position of the first and second locations where identity
30 parameters are changed. Accordingly, the network maneuvering described herein

provides a method for changing a mission plan or security model in a purely spontaneous manner based on a variety of factors, thereby increasing the security thereof.

Mission Plans

5 According to a preferred embodiment of the invention, the network maneuvering described herein is controlled in accordance with a mission plan. A mission plan is a schema that defines and controls maneuverability within the context of a network and at least one security model. As such, the mission plan can be represented as a data file that is communicated from the NAC **104** to each module
10 **105-107, 113-114** of FIG. 1. The mission plan is thereafter used by each module to control the manipulation of identity parameters and coordinate its activities with the actions of the other modules in the network.

 According to a preferred embodiment, the mission plan can be modified from time to time by a network administrator to update or change the way in which
15 the network maneuvers to thwart potential adversaries. As such, the mission plan provides a network administrator with a tool that facilitates complete control over the time, place and manner in which network maneuvering will occur within the network. Such update ability allows the network administrator to tailor the behavior of the computer network to the current operating conditions and more effectively thwart
20 adversary efforts to infiltrate the network. Multiple mission plans can be defined by a user and stored so that they are accessible to modules within the network. For example, the multiple mission plans can be stored at NAC **104** and communicated to modules as needed. Alternatively, a plurality of mission plans can be stored on each module and can be activated as necessary or desirable to maintain security of the
25 network. For example, if the network administrator determines or suspects that an adversary has discovered a current mission plan for a network, the administrator may wish to change the mission plan. Effective security procedures can also dictate that the mission plan be periodically changed.

 The process of creating a mission plan can begin by modeling the
30 computer network **100**. The creation of the model is facilitated by an NCSA

executing on a computer or server at the network command center. For example, in the embodiment shown in FIG. 1, the NCSA can execute on NAC **104**. The network model preferably includes information which defines data connections and/or relationships between various computing devices included in the computer network

5 **100**. The NCSA will provide a suitable interface which facilitates entry of such relationship data. According to one embodiment, the NCSA can facilitate entry of data into tables which can be used to define the mission plan. However, in a preferred embodiment, a GUI is used to facilitate this process.

Referring now to FIG. 3, the NCSA can include a network topography

10 model generator tool. The tool is used to assist the network administrator in defining the relationship between each of the various components of the networks. The network topography tool provides a workspace **300** in which an administrator can drag and drop network components **302**, by using a cursor **304**. The network administrator can also create data connections **306** between various network

15 components **302**. As part of this modeling process, the network administrator can provide network address information for the various network components, including the modules **105-107, 113, 114** of FIG. 1.

Once the network has been modeled, it can be saved and used by the network administrator to define the manner in which the various modules **105-107, 113, 114** behave and interact with one another. Referring now to FIG. 4, the NCSA

20 can generate a dialog box **400** of which can be used to further develop a mission plan. A drop-down menu **432** can be used to select the particular module (e.g., module **105** of FIG. 1) to which the settings in dialog box **400** are to be applied. Alternatively, the network administrator can use drop-down menu **432** to indicate that the settings in

25 dialog box **400** are intended to be applied to all modules within the network (e.g., by selecting the command "All" in the drop-down menu **432**). The process can continue by specifying whether a fixed set of identity parameters will always be modified in each of the modules, or whether the set of identity parameters that are manipulated shall be dynamically varied. If the selection or set of identity parameters that are to

30 be manipulated in the modules is intended to be dynamically varied, the network

administrator can mark check-box **401** to indicate that preference. If the check-box **401** is not marked, then the set of identity parameters to be varied is a fixed set that does not vary over time.

The dialog box **400** includes tabs **402**, **404**, **406** which allow a user to
5 select the particular identity parameter that he/she wants to work with for purposes of creating a mission plan. For purposes of this disclosure, the dialog box **400** facilitates dynamic variation of only three identity parameters. Specifically, these include the IP address, MAC address and port address. More or fewer identity parameters can be dynamically varied by providing additional tabs, but the three identity parameters
10 noted are sufficient to explain the inventive concepts. In FIG. 4, the user has selected the tab **402** to work with the IP address type of identity parameter. Within tab **402**, a variety of user interface controls **408-420** are provided for specifying the details relating to the dynamic variation of IP addresses within the selected module. More or fewer controls can be provided to facilitate the dynamic manipulation of the IP
15 address type, and the controls shown are merely provided to assist the reader in understanding the concept. In the example shown, the network administrator can enable dynamic variation of IP addresses by selecting (e.g., with a pointing device such as a mouse) the check-box **408** marked: "Enable IP Address Hopping". Similarly, the network administrator can indicate whether the source address,
20 destination address or both are to be varied. In this example, the source and destination address boxes **410**, **412** are both marked, indicating that both types of addresses are to be changed. The range of allowed values for the source and destination addresses can be specified by the administrator in list boxes **422**, **424**.

The particular pseudorandom process used to select false IP address values
25 is specified by selecting a pseudorandom process. This selection is specified in boxes **414**, **415**. Different pseudorandom processes can have different levels of complexity for variable degrees of true randomness, and the administrator can choose the process that best suits the needs of the computer network **100**.

Dialog box **400** also allows a network administrator to set the trigger type
30 to be used for the dynamic variation of the IP address identity parameter. In this

example, the user has selected box **416**, indicating that a time based trigger is to be used for determining when to transition to new false IP address values. Moreover, checkbox **418** has been selected to indicate that the time based trigger is to occur on a periodic basis. Slider **420** can be adjusted by the user to determine the frequency of the periodic time based trigger. In the example shown, the trigger frequency can be adjusted between six trigger occurrences per hour (trigger every ten minutes) and one hundred twenty trigger occurrences per hour (trigger every thirty seconds). In this example, selections are available for other types of triggers as well. For example, dialog box **402** includes check boxes **428**, **430** by which the network administrator can select an event-based trigger. Several different specific event types can be selected to form the basis for such event-based triggers (e.g., Event type 1, Event type 2, etc.). These event types can include the detection of: a packet having a particular origin or destination; a code word contained in a packet; secret or confidential information contained in a packet; a particular level of congestion; a particular traffic pattern; a particular protocol pattern; a particular entropy pattern; a security threat; an NBA of a particular level and/or type; and a particular number of NBAs currently being waged on a computer network. In FIG. 4, tabs **404** and **406** are similar to tab **402**, but the controls therein are tailored to the dynamic variation of the MAC address and port value rather than the IP address. Additional tabs could be provided for controlling the dynamic variation of other types of identity parameters.

The mission plan can also specify a plan for dynamically varying the location where identity parameters are modified. In some embodiments, this variable location feature is facilitated by controlling a sequence that defines when each module is in an active state or a bypass state. Accordingly, the mission plan advantageously includes some means of specifying this sequence. In some embodiments of the invention, this can involve the use of defined time intervals or time slots, which are separated by the occurrence of a trigger event.

Referring now to FIG. 5, a dialog box **500** can be provided by the NCSA to facilitate coordination and entry of location sequence and timing information. Dialog box **500** can include a control **502** for selecting a number of time slots **504**–

504_n which are to be included within a time epoch **506**. In the example illustrated, the network administrator has defined four time slots per timing epoch. The dialog box **500** can also include a table **503** which includes all modules in the computer network **100**. For each module listed, the table includes a graphical representation of available time slots **504₁-504₄** for one timing epoch **506**. Recall that dynamic control over the location where identity parameters are manipulated is determined by whether each module is in an active or bypass operating states. Accordingly, within the graphical user interface, the user can move a cursor **508** and make selections to specify whether a particular module is in an active or bypass mode during each time slot. In the example shown, module **105** is active during time slot **504₁** and **504₃**, but is in a bypass mode during time slots **504₂**, **504₄**. Conversely, module **113** is active during time slots **504₂**, **504₄**, but is in bypass mode during time slots **504₁** and **504₃**. With reference to FIG. 1, this means that manipulation of identity parameters occurs at a location associated with module **105** during time slots **504₁** and **504₃**, but occurs instead at module **113** during time slots **504₂**, **504₄**.

In the example shown in FIG. 5, the network administrator has elected to have module **114** always operate in an active mode (i.e., module **114** is active during all time slots). Accordingly, for data communications transmitted from client computer **101** to client computer **103**, data packets will alternately be manipulated in modules **105**, **113**, but will always be manipulated at module **114**. Finally, in this example, the network administrator has elected to maintain modules **106** and **107** in a bypass mode during time slots **504₁-504₄**. Accordingly, no manipulation of identity parameters will be performed at these modules during any of the defined time slots. Once the module timing has been defined in dialog box **500**, the network administrator can select the button **510** to store the changes as part of an updated mission plan. The mission plan can be saved in various formats. In some embodiments, the mission plan can be saved as a simple table or other type of defined data structure that can be used by each module for controlling the behavior of the module.

Distribution and Loading of Mission Plans

The distribution and loading of mission plans as disclosed herein will now be described in further detail. Referring once again to FIG. 1, it can be observed that the modules **105-107, 113, 114** are distributed throughout the computer network **100** at one or more locations. The modules are integrated within the communications pathways to intercept communications at such locations, perform the necessary manipulations, and forward data to other computing devices within the network. With the foregoing arrangement, any necessary maintenance of the modules described herein (e.g., maintenance to update a mission plan) will have the potential to disrupt network communications while the modules are replaced or reprogrammed. Such disruptions are undesirable in many situations where reliability and availability of network services is essential. For example, uninterrupted network operation can be essential for computer networks used by military, emergency services and businesses.

In order to ensure uninterrupted network operations, each module preferably has several operating states. These operating states include (1) an off state in which the module is powered down and does not process any packets, (2) an initialization state in which the module installs software scripts in accordance with the mission plan, (3) an active state in which data is processed in accordance with a current mission plan, and (4) a by-pass state in which packets can flow through the module as if the module was not present. The module is configured so that, when it is in the active state or the by-pass state, the module can receive and load an updated mission plan provided by a network administrator. The module operating states can be manually controlled by the network administrator by means of the NCSA executing, for example, on NAC **104**. For example, the user can select operating states for various modules through the use of a GUI control panel. Commands for controlling the operating states of the network are communicated over the computer network **100**, or can be communicated by any other suitable means. For example, a separate wired or wireless network (not shown) can be used for that purpose.

The mission plan can be loaded directly at the physical location of each module, or it can be communicated to the module from the NCSA. This concept is

illustrated in FIG. 6, which shows mission plans **604** being communicated from NCSA **602** to each of the modules **105-107, 113, 114** over a communication medium **606**. In the example shown, the NCSA software application is executing on NAC **104** operated by a network administrator. The communication medium can in some
5 embodiments include in-band signaling using computer network **100**. Alternatively, an out-of-band network (e.g., a separate wireless network) can be used as the communication medium **606** to communicate the updated mission plan from the NCSA to each module. As shown in FIG. 7, the NCSA can provide a dialog box **700** to facilitate selection of one of several mission plans **702**. Each of these mission plans
10 **702** can be stored on NAC **104**. The network administrator can select from one of the several mission plans **702**, after which they can activate a “Send Mission Plan” button **704**. Alternatively, a plurality of mission plans can be communicated to each module and stored there. In either scenario, the user can choose one of the defined mission plans to activate.

15 In response to the command to send the mission plan, the selected mission plan is communicated to the modules while they are in an active state in which they are configured for actively performing dynamic modification of identity parameters as described herein. Such an arrangement minimizes the time during which the network operates in the clear and without manipulating identity parameters. However, the
20 updated mission plan can also be communicated to the modules while they are in the by-pass mode, and this approach may be desirable in certain cases.

Once the mission plan is received by a module, it is automatically stored in a memory location within the module. Thereafter, the module can be caused to enter the by-pass state and, while still in that state, the module can load the data associated
25 with the new mission plan. This process of entering into the by-pass state and loading the new mission plan data can occur automatically in response to receipt of the mission plan, or can occur in response to a command from the NCSA software controlled by the network administrator. The new mission plan preferably includes changes in the way that identity parameter values are varied. Once the new mission
30 plan has been loaded, the modules **105-107, 113, and 114** can be transitioned from the

by-pass mode to the active mode in a synchronized way to ensure that data communication errors do not occur. The mission plan can specify a time when the modules are to return to the active mode, or the network administrator can use the NCSA to communicate a command to the various modules, directing them to enter
5 into the active mode. The foregoing process of updating a mission plan advantageously allows changes in network security procedures to occur without disrupting communication among the various computing devices attached to the computer network **100**.

The dynamic manipulation of various identity parameters at each module
10 **105, 106, 107, 113, and 114** is preferably controlled by the application software executing on each module **105-107, 113, 114**. However, the behavior of the application software is advantageously controlled by the mission plan.

Referring now to FIG. 8, there is provided a flowchart which summarizes the operation of each module **105-107, 113, 114**. To avoid confusion, the process **800**
15 is described with respect to communications in a single direction. For example, in the case of module **105**, the single direction could involve data transmitted from client computer **101** to hub **108**. In practice however, it is preferred that modules **105-107, 113, 114** operate bi-directionally. The process begins at step **802** when the module is powered up and continues to step **804** where module application software is initialized
20 for executing the methods described herein. In step **806**, a mission plan is loaded from a memory location within the module. At this point, the module is ready to begin processing data and proceeds to do so at step **808**, where it accesses a data packet from an input data buffer of the module. In step **810**, the module checks to determine if it is in a bypass mode of operation. If so, the data packet accessed in step
25 **808** is retransmitted in step **812** without any modification of the data packet. If the module is not in bypass mode, then it must be in its active mode of operation and continues on to step **814**. In step **814**, the module reads the data packet to determine the identity of a source node from which the data packet originated. In step **816**, it examines the packet to determine if the source node is valid. The specified source
30 node can be compared to a list of valid nodes to determine if the specified source node

is currently valid. If it is not a valid node then the packet is discarded in step **818**. In step **820**, the process checks to determine if a trigger event occurred. The occurrence of a trigger event will influence the selection of false identify values to use.

Accordingly, in step **822**, the module determines the false identify values to use based
5 on one or more of the trigger information, clock time and mission plan. The module then continues to step **826** where it manipulates identity parameters of the data packet. Once manipulations are complete, the data packet is re-transmitted to an adjacent node from the output port of the module. In step **830**, a determination is made as to whether the module has been commanded to power down. If so, the process ends at
10 step **832**. In step **808**, the process continues and the next data packet is accessed from the module's input data buffer.

Referring now to FIG. 9, there is provided a flowchart which summarizes the methods described herein for managing a dynamic computer network. The process begins in step **902** and continues to step **904**, where a network model is
15 created (e.g., as shown and described in relation to FIG. 3). In step **906**, a determination is made as to whether a new mission plan is to be created. If so, a new mission plan is created in step **908** and the process continues to step **910**, where the new mission plan is selected. Alternatively, if in step **906** a desired mission plan has already been created, then the method can continue directly to step **910** where an
20 existing mission plan is selected. In step **912**, the mission plan is communicated to the modules (e.g., modules **105-107, 113, 114** of FIG. 1), where the mission plan is stored in a memory location. When the network administrator is ready to implement the new mission model, a command is sent in step **914** which causes the modules to enter a standby mode as described herein. While the modules are in this standby
25 mode, the mission plan is loaded at step **916**. Loading of the mission plan occurs at each module so that the mission plan can be used to control the operations of an application software executing on the module. In particular, the mission plan is used to control the way in which the application software performs dynamic manipulations of identity parameters. In step **918**, the mission modules are again caused to enter
30 into an active operational mode in which each mission module performs

manipulations of identity parameters in accordance with the mission plan. Steps **914**, **916** and **918** can occur in response to specific commands sent from a network administrator, or can occur automatically at each module in response to receiving the mission plan in step **912**. After step **918**, the modules continue performing processing in accordance with the mission plan which has been loaded. In step **920**, the process continues by checking to determine if the user has indicated a desire to change the mission plan; if so, the process returns to step **906**, where it continues as described above. If there is no indication that the user or network administrator wishes to change an existing mission plan, then the process determines in step **922** whether it has been instructed to terminate. If so, the process terminates in step **924**. If no termination instruction is received, the process returns to step **920** and continues.

Referring now to FIG. 10, there is provided a block diagram which shows a computer architecture of an exemplary module **1000** which can be used for performing the manipulation of identity parameters described herein. The module **1000** includes a processor **1012** (such as a Central Processing Unit (“CPU”)), a main memory **1020** and a static memory **1018**, which communicate with each other via a bus **1022**. The module **1000** can further include a display unit **1002**, such as a Liquid Crystal Display (“LCD”) to indicate the status of the module. The module **1000** can also include one or more network interface devices **1016**, **1017** which allow the module to receive and transmit data concurrently on two separate data lines. The two network interface ports facilitate the arrangement shown in FIG. 1, where each module is configured to concurrently intercept and re-transmit data packets received from two separate computing devices on the network.

The main memory **1020** includes a computer-readable storage medium **1010** on which is stored one or more sets of instructions **1008** (e.g., software code) configured to implement one or more of the methodologies, procedures, or functions described herein. The instructions **1008** can also reside, completely or at least partially, within the static memory **1018**, and/or within the processor **1012** during execution thereof by the module. The static memory **1018** and the processor **1012** also can constitute machine-readable media. In the various embodiments of the

present invention a network interface device **1016** connected to a network environment communicates over the network using the instructions **1008**.

The instructions **1008** cause the module **1000** to act as a translator of identity parameters between those of a packet-based static network and those of a packet-based MTT enabled network. A conventional protocol stack for the packet-based static network is provided in FIG. 11. According to the embodiment shown in FIG. 11, the protocol stack **1100** includes five layers **1102**, **1104**, **1106**, **1108**, **1110** specifying particular functions of nodes within the packet-based static network. Still, the invention is not limited in this regard. The protocol stack **1100** can include any number of layers in accordance with a particular packet-based static network application. For example, if an Open System Interconnection (“OSI”) protocol stack is employed by the static network then the protocol stack **1100** can further include a session layer and a presentation layer.

Referring again to FIG. 11, the protocol stack **1100** provides a framework illustrating how information is passed from a software application installed in a first node of the static network (e.g., a client computer) to a software application installed in a second node of the static network (e.g., a client computer). The protocol stack **1100** is well known to persons skilled in the art. Thus, the protocol stack **1100** will not be described in detail herein. However, a brief discussion is provided below to assist a reader in understanding the identity parameter translation which is performed at least by the modules **105-108**, **114** of FIG. 1.

As shown in FIG. 11, the protocol stack **1100** is comprised of a physical layer **1102**, a data link layer **1104**, a network layer **1106**, a transport layer **1108**, and an application layer **1110**. The physical layer **1102** is comprised of firmware and/or hardware configured to send and receive data through a network. The data link layer **1104** provides transmission protocols for transferring data between network nodes. Such transmission protocols can include an Ethernet protocol (or an IEEE 802.3 protocol), a point-to-point protocol, an IEEE 802.11 protocol, an IEEE 802.15 protocol, an IEEE 802.16 protocol, and other such protocols.

The data link layer **1104** can be comprised of two (2) sub-layers, namely a Logic Link Control (“LLC”) layer **1114** and a Media Access Control (“MAC”) layer **1112**. The LLC layer **1114** is comprised of firmware and/or hardware configured to multiplex protocols prior to being transmitted over the MAC layer **1112** and to de-
5 multiplex protocols subsequent to being transmitted and upon receipt. The LLC layer **1114** is also comprised of firmware and/or hardware configured to provide flow control of packets, detection of packets, and retransmission of dropped packets.

The MAC layer **1112** is comprised of firmware and/or hardware configured to determine when to transmit communications and when to receive
10 communications. In this regard, the MAC layer **1112** performs actions involving coordinating access to a shared radio channel and utilizing protocols that enhance communications over a wireless link. The term "protocol" as used herein refers to a set of rules defining how information is exchanged between network nodes. Such network nodes include, but are not limited to, the client computers, servers, routers,
15 switches and bridges. The MAC layer **1112** provides transmission protocols for transferring data between network nodes. Such transmission protocols include MAC protocols. MAC protocols ensure that signals sent from different nodes across the same channel do not collide.

The network layer **1106** is comprised of firmware configured to transfer
20 data from one node to another node. In this regard, the network layer **1106** provides protocols for transferring data from one node to another node. The transmission protocols include routing protocols and forwarding protocols. Such transmission protocols include internet protocols, such as a version four of the internet protocol (“IPv4”), a version six of the internet protocol (“IPv6”), and internet security
25 protocols (“IP Layer Security”).

The transport layer **1108** is comprised of firmware configured to communicate data between end systems. In this regard, the transport layer **1108** provides transport protocols for transmission of data between end systems. Such transport protocols include a Transmission Control Protocol (“TCP”) and a User
30 Datagram Protocol (“UDP”). The application layer **1110** is generally implemented

only in firmware. The application layer **1110** provides signaling protocols for end-user applications, such as authentication applications, data syntax applications, quality of service applications, and end-user applications.

Referring now to FIG. 12, there is provided a block diagram of a
5 conventional packet **1200** of a static network. The packet **1200** is comprised of a preamble **1202**, a physical layer protocol header **1204**, a MAC layer protocol header **1206**, an LLC layer protocol header **1208**, a network layer protocol header **1210**, and a transport layer protocol header **1212**. The packet **1200** is also comprised of an application layer header **1214**, an application data **1216**, and a Frame Check Sequence
10 ("FCS") **1218**. The phrase "frame check sequence", as used herein, refers to extra checksum characters added to a packet or a frame in a communication protocol for error detection and correction. Each of the listed components of the packet **1200** are well known to persons skilled in the art and are well defined in open industry standards of the Institute of Electrical and Electronics Engineers ("IEEE") standard
15 for local and metropolitan area networks and Internet Engineering Task Force ("IETF"). Thus, such components will not be described in detail herein.

However, it should be appreciated that the application data **1216** can be signaling protocol data, user data, or management data. The user data can include voice data, video data, or the like. It should also be appreciated that the application
20 data **1216** is encapsulated between the application layer header **1214** and the FCS **1218**. The application layer header **1214** is encapsulated between the transport layer protocol header **1212** and the application data **1216**. Similarly, the transport layer protocol header **1212** is encapsulated between the network layer protocol header **1210** and the application layer header **1214**. Likewise, the network layer protocol header
25 **1210** is encapsulated between the LLC layer protocol header **1208** and transport layer protocol header **1212**. The LLC layer protocol header **1208** is encapsulated between the MAC layer protocol header **1206** and the network layer protocol header **1210**. The MAC layer protocol header **1206** is encapsulated between the physical layer protocol header **1204** and the LLC layer protocol header **1208**. The physical layer

protocol header **1204** is encapsulated between the preamble **1202** and the MAC layer protocol header **1206**.

The transport layer protocol header **1212** comprises source and destination port numbers **1220**. A port is an application-specific software construct serving as a communications endpoint in a computer's operating system. A port is identified for
5 each IP address and protocol by a sixteen bit number (i.e., a port number **1220**).

The transport layer protocol header **1212** also comprises a TCP sequence number **1222**. Two client computers communicating with each other on opposite sides of a TCP session will each maintain a TCP sequence number **1222**. The TCP
10 sequence number **1222** allows each computer to track how much data it has communicated. The TCP sequence number is included in the TCP header portion of each packet which is communicated during the session. At the initiation of a TCP session, the initial sequence number value is randomly selected.

The network layer protocol header **1210** comprises source and destination
15 IP addresses **1224**. An IP address **1224** is a numerical identifier assigned to a computing device participating in a computer network where the network uses the well known internet protocol for communication. The IP address **1224** can be a thirty two bit number in an IPv4 system or one hundred twenty eight bit number in an IPv6 system. The IP address **1224** is a binary number, but is usually stored in a text file
20 and displayed in a human-readable notation (e.g., 175.18.252.1. for IPv4 systems and 2003:db6:0:1234:0:469:6:1 for IPv6).

In some embodiments, each IP address **1224** can be thought of as a single identity parameter. However, an IP address **1224** is generally defined as including at least two parts which include a network prefix **1228** and a host number **1230**. The
25 network prefix **1228** identifies a network to which a data packet **1200** is to be communicated. The host number **1230** identifies the particular node within a Local Area Network ("LAN"). A sub-network (sometimes referred to as a subnet) is a logical portion of an IP network. Where a network is divided into two or more sub-networks, a portion of the host number **1230** of the IP address **1224** is used to specify
30 a subnet number **1232**. For purposes of the present invention, the network prefix

1228, the subnet number **1232** and the host number **1230** can each be considered to be a separate identity parameter. Since a source IP address and a destination IP address is contained in the network layer protocol header **1210**, there are a total of six different identity parameters in the header **1210**.

5 The MAC layer protocol header **1206** comprises a MAC Address **1226**. A MAC address **1226** is a unique value assigned to a network interface device by a manufacturer and stored in an onboard ROM. The MAC address **1226** can include a forty-eight bit number or a sixty-four bit number depending on the protocol employed by the MAC layer **1112** of the protocol stack **1100**.

10 The MTT enabled network employs protocols of an MTT protocol stack. A schematic illustration of an exemplary MTT protocol stack **1300** is provided in FIG. 13. As shown in FIG. 13, the MTT protocol stack **1300** comprises five layers **1302-1314** specifying particular functions of nodes within the MTT enable network. Notably, some of the layers **1302**, **1310**, **1314** are the same as those **1102**, **1110**, **1114**
15 of the protocol stack **1100**. As such, the description provided above in relation to these layers **1102**, **1110**, **1114** is sufficient for understanding layers **1302**, **1310**, **1314** of FIG. 13. However, the MTT protocol stack **1300** comprises layers **1304-1308** which are different than those **1104-1108** of protocol stack **1100**. As such, a brief discussion of these layers will be provided below.

20 The MTT data link layer **1304** can be comprised of two (2) sub-layers, namely an LLC layer **1314** and an MTT MAC layer **1312**. The LLC layer **1314** is the same as or substantially similar to the LLC layer **1114** of FIG. 11. As such, the description provided above in relation to layer **1114** is sufficient for understanding layer **1314**. The MTT MAC layer **1312** is different than the MAC layer **1112** of FIG.
25 11. In this regard, it should be understood that the MAC layer **1112** employs a static MAC address **1226** for each network interface device. In contrast, the MTT MAC layer **1312** employs a non-static MAC address (e.g., MTT MAC address **1426** of FIG. 14) for each network interface device. The non-static MAC address is dynamically variable. For example, the non-static MAC address can be randomly or pseudo-
30 randomly changed during operation of the MTT enabled network.

The MTT network layer **1306** is different than the network layer **1106** of FIG. 11. In this regard, it should be understood that the network layer **1106** employs static IP addresses **1224**. In contrast, the IP addresses (e.g., MTT IP addresses **1424** of FIG. 14) of the MTT network layer **1306** are non-static (i.e., they can be

5 dynamically varied during operations of the MTT enabled network). For example, an IP address number can be changed in accordance with a pseudo-random process.

The MTT transport layer **1308** is different than the transport layer **1108** of FIG. 11. The transport layer **1108** employs static port numbers **1220** and static TCP sequence numbers **1222**. In contrast, the port numbers (e.g., numbers **1420** of FIG.

10 14) and TCP sequence numbers (e.g., numbers **1422** of FIG. 14) employed by the MTT transport layer **1308** are non-static. In this regard, it should be understood that each of the non-static port numbers and sequence numbers can be changed in accordance with a random or pseudo-random process.

Referring now to FIG. 14, there is provided a schematic illustration of an

15 exemplary MTT packet **1400** of the MTT enabled network. The MTT packet **1400** is comprised of a preamble **1402**, a physical layer protocol header **1404**, an MTT MAC layer protocol header **1406**, an LLC layer protocol header **1408**, an MTT network layer protocol header **1410**, an MTT transport layer protocol header **1412**, an application layer header **1414**, application data **1416** and an FCS **1418**. Portions

20 **1402**, **1404**, **1408**, **1414**, **1416**, **1418** of the MTT packet **1400** are the same as or substantially similar to portions **1202**, **1204**, **1208**, **1214**, **1216**, **1218** of FIG. 12. As such, the description provided above in relation to portions **1202**, **1204**, **1208**, **1214**, **1216**, **1218** is sufficient for understanding portions **1402**, **1404**, **1408**, **1414**, **1416**, **1418** of the MTT packet **1400**. However, portions **1406**, **1410**, **1412** are different than

25 portions **1206**, **1210**, **1212** of FIG. 12. As such, each of the portions **1406**, **1410**, **1412** will be described herein.

The MTT transport layer protocol header **1412** comprises MTT source and destination port numbers **1420**. A port is an application-specific software construct serving as a communications endpoint in a computer's operating system. A port is

30 identified for each IP address and protocol by a sixteen bit number. The sixteen bit

number is referred to in relation to an MTT enabled network as an MTT port number **1420**. Notably, each MTT port number **1420** is a non-static number (i.e., it can be changed by a module in accordance with a random or pseudo-random process). The MTT transport layer protocol header **1412** also comprises an MTT TCP sequence number **1422**. The MTT TCP sequence number **1422** is a non-static number. Manipulation of the MTT port number **1420** and the MTT TCP sequence number **1422** can be accomplished by simply modifying the TCP header information to change values thereof.

The MTT network layer protocol header **1410** comprises source and destination MTT IP addresses **1424**. A value of each MTT IP address **1424** can be dynamically varied in accordance with a random or pseudo-random process. Each MTT IP address **1424** comprises an MTT prefix **1428** and an MTT host number **1430**. Each of these components **1428**, **1430** can also be dynamically varied during operation of an MTT enabled network. Manipulation of the MTT IP addresses **1424**, the MTT prefix **1428** and the MTT host number **1430** can be achieved by simply modifying the IP header information of the MTT network layer protocol header **1410**.

[0100] The MTT MAC layer protocol header **1406** comprises an MTT MAC Address **1426**. The MTT MAC address **1426** can be dynamically varied during operation of the MTT enabled network. Manipulation of the MTT MAC address **1426** can be achieved by simply modifying an Ethernet header information of the MTT MAC layer protocol header **1406**.

Referring now to FIG. 15, there is provided a schematic illustration that is useful for understanding operations of the module **1000** when it is implemented as hardware. As shown in FIG. 15, the module **1000** performs operations in accordance with both protocol stacks **1200**, **1400**. In this regard, the module **1000** is configured to communicate MTT packets **1400** to and from an MTT enabled network node **1502** (e.g., a node **104-109**, **111** or **112** of FIG. 1). The module **1000** is also configured to communicate conventional packets **1200** to and from a static enabled network node **1506** (e.g., a node **101-103**, **110** or **115** of FIG. 1). The module **1000** is further configured to convert conventional packets **1200** into MTT enabled packets **1400**, and

vice versa. This packet conversion is achieved via an Identity Parameter Translation (“IPT”) **1504**. A process for achieving an IPT generally involves: de-encapsulating and re-encapsulating application layer portions **1214, 1216, 1414, 1416** of the packets **1200, 1400**. Methods for encapsulating and de-encapsulating packets are well known
5 in the art, and therefore are not be described herein. Any known method or to be known method for encapsulating and de-encapsulating packets can be used with the present invention without limitation. Examples of such processes for achieving an IPT are illustrated in FIGS. 16 and 17.

As shown in FIG. 16, an exemplary process **1600** begins at step **1602** and
10 continues with step **1604**. In step **1604**, the application layer portions **1214, 1216** of the conventional packet **1200** are de-encapsulated by removing the preamble **1202** and headers **1204-1212** from the packet. In a next step **1606**, the packet components **1402-1414** are generated in accordance with the protocols of the MTT protocol stack **1300**. Thereafter, step **1608** is performed where the application layer portions **1214,**
15 **1216** are re-encapsulated so as to form an MTT packet **1400**. The re-encapsulation is achieved by appending the packet components **1402-1414** thereto. Upon completing step **1608**, the process **1600** ends or other processing is performed.

As shown in FIG. 17, an exemplary process **1700** begins at step **1702** and continues with step **1704**. In step **1704**, the application layer portions **1414, 1416** of
20 an MTT packet **1400** are de-encapsulated by removing the preamble **1402** and headers **1404-1412** from the packet **1400**. Next in step **1706**, the packet components **1204-1214** are generated in accordance with the protocols of the conventional protocol stack **1200**. In a next step **1708**, the application layer portions **1414, 1416** are re-encapsulated by appending the preamble **1202** and headers **1204-1214** thereto so as to
25 form a conventional packet **1200**.

Referring now to FIG. 18, there is shown an exemplary NAC **104** in accordance with the inventive arrangements. The NAC **104** can comprise various types of computing systems and devices, including a server computer, a client user computer, a Personal Computer (“PC”), a tablet PC, a laptop computer, a desktop
30 computer, a control system or any other device capable of executing a set of

instructions (sequential or otherwise) that specifies actions to be taken by that device. Further, while a single computer is illustrated in FIG. 18, the phrase “NAC” shall be understood to include any collection of computing devices that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the
5 methodologies discussed herein.

Referring now to FIG. 18, the NAC **104** includes a processor **1812** (such as a CPU), a disk drive unit **1806**, a main memory **1820** and a static memory **1818**, which communicate with each other via a bus **1822**. The NAC **104** can further include a display unit **1802**, such as a video display (e.g., an LCD), a flat panel, a
10 solid state display, or a Cathode Ray Tube (“CRT”). The NAC **104** can include a user input device **1804** (e.g., a keyboard), a cursor control device **1814** (e.g., a mouse) and a network interface device **1816**.

The disk drive unit **1806** includes a computer-readable storage medium **1810** on which is stored one or more sets of instructions **1808** (e.g., software code)
15 configured to implement one or more of the methodologies, procedures, or functions described herein. The instructions **1808** can also reside, completely or at least partially, within the main memory **1820**, the static memory **1818**, and/or within the processor **1812** during execution thereof. The main memory **1820** and the processor **1812** also can constitute machine-readable media.

Those skilled in the art will appreciate that the module architecture
20 illustrated in FIGS. 10-17 and the NAC architecture in FIG. 18, each represent merely one possible example of a computing device that can be used respectively for performing the methods described herein. However, the invention is not limited in this regard and any other suitable computing device architecture can also be used
25 without limitation. Dedicated hardware implementations including, but not limited to, application-specific integrated circuits, programmable logic arrays, and other hardware devices can likewise be constructed to implement the methods described herein. Applications that can include the apparatus and systems of various
30 embodiments broadly include a variety of electronic and computer systems. Some embodiments may implement functions in two or more specific interconnected

hardware devices with related control and data signals communicated between and through the modules, or as portions of an application-specific integrated circuit. Thus, the exemplary system is applicable to software, firmware, and hardware implementations.

5 In accordance with various embodiments of the present invention, the methods described herein are stored as software programs in a computer-readable storage medium and are configured for running on a computer processor. Furthermore, software implementations can include, but are not limited to, distributed processing, component/object distributed processing, parallel processing, virtual
10 machine processing, which can also be constructed to implement the methods described herein.

 While the computer-readable storage medium **1010**, **1810** is shown in FIGS. 10 and 18 to be a single storage medium, the term "computer-readable storage medium" should be taken to include a single medium or multiple media (e.g., a
15 centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "computer-readable storage medium" shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present disclosure.

20 The term "computer-readable medium" shall accordingly be taken to include, but is not be limited to, solid-state memories such as a memory card or other package that houses one or more read-only (non-volatile) memories, random access memories, or other re-writable (volatile) memories; magneto-optical or optical mediums such as a disk or tape. Accordingly, the disclosure is considered to include
25 any one or more of a computer-readable medium as listed herein and to include recognized equivalents and successor media, in which the software implementations herein are stored.

Shadow Networking

 In some scenarios, the above described computer network may still be
30 vulnerable to malicious attacks. As such, the present invention provides a way to

detect, deter and delay attackers of the computer network using the above described MTT technology. The present invention employs shadow networking to detect, deter and delay attackers. In general, shadow networking involves using identity parameter values to detect suspect packets indicating that there is a possible malicious attack
5 being waged on the computer network. Once a possible malicious attack has been detected, one or more actions can be taken so as to deter and/or delay the attacker. Such actions can include, but are not limited to, analyzing a suspect packet to determine the source thereof, tracking the sources of suspect packets, logging information relating to suspect packets, dropping suspect packets, and/or sending
10 responses to the sources of the suspect packets. The listed actions can be performed for purposes of (a) making it hard for attackers to determine the locations and/or types of network devices contained in the computer network, and/or (b) hiding the fact that attack attempts have been detected.

The particularities of shadow networking will now be described in relation
15 to FIGS. 19-21. FIG. 19 is a schematic illustration of two exemplary mission plans **1900, 1902**. As shown in FIG. 19, each mission plan contains a mission model **1904, 1906** describing how network devices are to act in accordance with MTT technology and shadow networking technology. More particularly, each mission model **1904, 1906** specifies that a value for at least one identity parameter (e.g., identity parameter
20 **1420, 1422, 1424, 1426, 1428** and/or **1432** of FIG. 14) of at least one network device **ND₁, ND₂** (e.g., network device **105-109, 111**, and/or **112** of FIG. 1) is to be periodically changed during the period of time when the respective mission plan **1900, 1902** is employed by the computer network (e.g., computer network **100** of FIG. 1). For example, each mission model **1904, 1906** specifies that values for IP
25 addresses of two network devices are to be periodically changed when a respective mission plan **1900, 1902** is employed by the computer network. Embodiments of the present invention are not limited in this regard. The mission models **1904, 1906** can indicate that values for different identity parameters are to be periodically changed when the mission plans **1900, 1902** are employed by the computer network.

As shown in FIG. 19, each mission model **1904**, **1906** indicates that a true value and a plurality of false values are assigned to the identity parameter of a network device **ND₁**, **ND₂**. For example, each mission model **1904**, **1906** indicates that the IP address of each network device **ND₁**, **ND₂** has nine possible values **TV**, **FV₁**, **FV₂**, **FV₃**, **FV₄**, **FV₅**, **FV₆**, **FV₇**, **FV₈**. The nine possible values for the IP address of network device **ND₁** include a true value **TV₁**=175.18.252.1 and false values **FV₁₋₁**=176.18.252.1, **FV₁₋₂**=175.18.252.2, **FV₁₋₃**=176.18.253.1, **FV₁₋₄**=177.18.252.2, **FV₁₋₅**=178.18.254.1, **FV₁₋₆**=179.20.252.3, **FV₁₋₇**=177.18.262.5, **FV₁₋₈**=179.12.263.7. The nine possible values for the IP address of network device **ND₂** include a true value **TV₂**=172.14.236.7 and false values **FV₂₋₁**=176.18.252.1, **FV₂₋₂**=175.18.254.2, **FV₂₋₃**=179.20.252.3, **FV₂₋₄**=179.12.263.7, **FV₂₋₅**=172.16.242.9, **FV₂₋₆**=179.18.251.3, **FV₂₋₇**=170.20.242.1, **FV₂₋₈**=179.14.262.1.

As evident from the above statements, the network devices **ND₁**, **ND₂** have some of the same false values assigned to their IP addresses (i.e., 176.18.252.1, 178.18.254.1, 179.20.252.3, 179.12.263.7). These false values are referred to herein as common false values. Notably, the common false values are not active (or not used) for (a) network device **ND₁** during the period of time when the mission plan **1900** is employed thereby and (b) network device **ND₂** during the period of time when the mission plan **1902** is employed thereby. As such, the common false values constitute “empty” IDentity Parameter Values (“IDPVs”) for (a) network device **ND₁** when the mission plan **1900** is employed thereby and (b) network device **ND₂** when the mission plan **1902** is employed thereby. In contrast, the common false values are active (or used) for (a) network device **ND₂** during the period of time when the mission plan **1900** is employed thereby and (b) network device **ND₁** during the period of time when the mission plan **1902** is employed thereby. Accordingly, the common false values constitute “active” IDPVs for (a) network device **ND₂** when the mission plan **1900** is employed thereby and (b) network device **ND₁** when the mission plan **1902** is employed thereby.

Although FIG. 19 indicates that all of the “empty” IDPVs of mission plan **1904** are “active” IDPVs in mission plan **1906**, embodiments of the present invention

are not limited in this regard. For example, the mission models **1904**, **1906** may specify that at least one false value constitutes an “empty” IDPV for a particular network device **ND₁** or **ND₂** when either mission plan **1900**, **1902** is employed thereby.

5 Each mission model **1904**, **1906** also specifies a timeslot in which the “active” IDPVs for the network devices are to be used. For example, mission model **1904** specifies that: false value **FV_{1,3}** for the IP address of network device **ND₁** is to be used during a second timeslot **TS₁**; true value **TV₁** for the IP address of network device **ND₁** is to be used during a second timeslot **TS₂**; false value **FV_{1,7}** for the IP
10 address of network device **ND₁** is to be used during a second timeslot **TS₃**; false value **FV_{1,4}** for the IP address of network device **ND₁** is to be used during a second timeslot **TS₄**; and false value **FV_{1,2}** for the IP address of network device **ND₁** is to be used during a second timeslot **TS₅**. Mission model **1904** also specifies that: false value **FV_{2,1}** for the IP address of network device **ND₂** is to be used during a second timeslot
15 **TS₁**; false value **FV_{2,2}** for the IP address of network device **ND₂** is to be used during a second timeslot **TS₂**; false value **FV_{2,3}** for the IP address of network device **ND₃** is to be used during a second timeslot **TS₃**; false value **FV_{2,4}** for the IP address of network device **ND₂** is to be used during a second timeslot **TS₄**; and true value **TV₂** for the IP address of network device **ND₂** is to be used during a second timeslot **TS₅**.

20 Each mission plan **1904**, **1906** further specifies shadow networking activities that are to be performed by modules (e.g., modules **105-107**, **113** and **114** of FIG. 1) of the computer network during operations thereof. For example, the mission plan **1904** indicates that each module is to analyze every packet received thereat to determine if it contains an “empty” IDPV or an “old (or shadow)” IDPV for the IP
25 address associated with network device **ND₁** and/or network device **ND₂**. If the module determines that a received packet comprises an “empty” IDPV for the IP address associated with the network device **ND₁** and/or network device **ND₂**, then the packet is dropped and forwarded to an intrusion analyzer so that information thereof can be logged. In contrast, if the module determines that a received packet comprises
30 an “old (or shadow)” IDPV for the IP address associated with network device **ND₁**

and/or network device **ND₂**, then the packet is forwarded to an intrusion analyzer so that information thereof can be logged and a response can be sent to the source thereof. “Old (or Shadow)” IPDVs will be described in more detail below in relation to FIGS. 20A-20E.

5 Referring now to FIGS. 20A-20E, there are provided schematic illustrations that are useful for understanding what constitutes “old (or shadow)” IPDVs. A time structure **2000** for mission plan **1900** is provided in FIG. 20A. As shown in FIG. 20A, mission plan **1900** is employed by network devices **ND₁**, **ND₂** during time frames **2020**, **2022**. A time structure **2002** for mission plan **1902** is provided in FIG. 20B. As shown in FIG. 20B, mission plan **1902** is employed by network devices **ND₁**, **ND₂** during time frames **2024**, **2026**. A time structure **2004** for a transition from mission plan **1900** to mission plan **1902** is provided in FIG. 20C. As shown in FIG. 20C, the mission plan transition occurs at the end of time frame **2022** in response to a trigger event **2006**. Trigger event **2006** includes any one of the
10 proactive or reactive trigger events described above.
15

Although FIGS. 20A-20C indicate that each mission plan **1900**, **1902** is employed for two consecutive time frames, embodiments to the present invention are not limited in this regard. Each mission plan **1900**, **1902** can be employed for a portion of a time frame, one time frame, or two or more consecutive time frames
20 during the period of time between two trigger events. Also, the mission plans **1900**, **1902** can be employed for a different number of consecutive or non-consecutive time frames in accordance with the occurrences of trigger events.

FIG. 20D provides more detailed schematic illustrations of time frames **2020**, **2022** shown in FIG. 20A. Similarly, FIG. 20E provides more detailed
25 schematic illustrations of time frames **2024**, **2026** shown in FIG. 20B. The contents of FIG. 20D will be described in detail below. However, the contents of FIG. 20E will not be described in detail below since the discussion of FIG. 20D is sufficient for understanding FIG. 20E.

As shown in FIG. 20D, each time frame **2020**, **2022** comprises a plurality
30 of timeslots **TS₁**, **TS₂**, **TS₃**, **TS₄**, **TS₅**. Each timeslot **TS₁**, **TS₂**, **TS₃**, **TS₄**, **TS₅** has the

same static duration, the same variable duration, or different durations. Each timeslot **TS₁, TS₂, TS₃, TS₄, TS₅** specifies a time period when a given “active” IDPV associated with a network device **ND₁, ND₂** in mission plan **1900** is used for an identity parameter value in packets communicated thereto and therefrom.

5 FIG. 20D also shows that timeslots **TS₁, TS₂, TS₃, TS₄, TS₅** of each time frame **2020, 2022** has sets of “shadow” IDPVs associated therewith for each network device **ND₁, ND₂**. Each set of “shadow” IDPVs includes the IDPVs which: (a) are defined in mission plan **1900** as “active” IDPVs for a respective network device; (b) were previously used as an identity parameter value in packets communicated to and
10 from the respective network device **ND₁** or **ND₂**; and (c) are not used during a current timeslot **TS₁, TS₂, TS₃, TS₄** or **TS₅** as said identity parameter value. For example, time frame **2020** is the first time frame in which mission plan **1900** is used. As such, during timeslot **TS₁** of time frame **2020**, there are no “shadow” identities for network devices **ND₁** and **ND₂**. In timeslot **TS₂** of time frame **2020**, the “shadow” identities
15 for network devices **ND₁, ND₂** comprise **FV₁₋₃, FV₂₋₁**, respectively. In timeslot **TS₃** of time frame **2020**, the “shadow” identities for network devices **ND₁, ND₂** comprise **TV₁, FV₁₋₃, FV₂₋₁, FV₂₋₂**, respectively. In timeslot **TS₄** of time frame **2020**, the “shadow” identities for network devices **ND₁, ND₂** comprise **TV₁, FV₁₋₃, FV₁₋₇, FV₂₋₁, FV₂₋₂, FV₂₋₃**, respectively. In timeslot **TS₅** of time frame **2020**, the “shadow”
20 identities for network devices **ND₁, ND₂** comprise **TV₁, FV₁₋₃, FV₁₋₄, FV₁₋₇, FV₂₋₁, FV₂₋₂, FV₂₋₃, FV₂₋₄**, respectively. In timeslot **TS₁** of time frame **2022**, the “shadow” IDPVs for network devices **ND₁, ND₂** comprise **TV₁, FV₁₋₂, FV₁₋₄, FV₁₋₇, TV₂, FV₂₋₂, FV₂₋₃, FV₂₋₄**, and so on. These “shadow” identities can be used by the modules during timeslot **TS₁** to detect a possible malicious attack on the computer network
25 (e.g., computer network **100** of FIG. 1). The manner in which the “shadow” identities are used to detect possible malicious attacks will become more evident as the discussion progresses.

Notably, sliding window intervals **2050, 2052, 2054, 2056** of FIG. 20D may be provided to account for the time (e.g., three seconds) it takes packets to travel
30 from a source device residing at a first location (e.g., the United States) to a

destination device located at a second remote location (e.g., the United Kingdom). In some embodiments, the sliding window intervals can be a feature which may be disabled or enable as desired. During each sliding window interval, the modules can use the “shadow” IDPVs for a previous timeslot and the “shadow” IDPVs for the

5 current timeslot for purposes of detecting a possible malicious attack. For example, during sliding window interval **2050**, the modules use the “shadow” IDPVs for timeslot **TS₁** and the “shadow” IDPVs for timeslot **TS₂** for purposes of detecting a possible malicious attack. During sliding window interval **2052**, the modules use the “shadow” IDPVs for timeslot **TS₂** and the “shadow” IDPVs for timeslot **TS₃** for

10 purposes of detecting a possible malicious attack. During sliding window interval **2054**, the modules use the “shadow” IDPVs for timeslot **TS₃** and the “shadow” IDPVs for timeslot **TS₄** for purposes of detecting a possible malicious attack. During sliding window interval **2056**, the modules use the “shadow” IDPVs for timeslot **TS₄** and the “shadow” IDPVs for timeslot **TS₅** for purposes of detecting a possible malicious

15 attack.

As shown in FIG. 20D, the sliding window intervals **2050**, **2052**, **2054**, **2056** have the same or substantially similar durations. Embodiments of the present invention are not limited in this regard. The sliding window intervals **2050**, **2052**, **2054**, **2056** can have different durations. Also, each of the sliding window intervals

20 **2050**, **2052**, **2054**, **2056** can have any duration that is less than a duration of each timeslot **TS₁**, **TS₂**, **TS₃**, **TS₄**, **TS₅**.

Referring now to FIG. 21, there is provided a flow diagram of an exemplary method **2100** for shadow networking. Method **2100** begins with step **2102** and continues with step **2104**. In step **2104**, a packet is received at an MTT enabled device (e.g., module **105-107**, **113** or **114** of FIG. 1). At the MTT enabled device, the

25 packet is analyzed to determine whether all of the identity parameters thereof have their “correct” values for the current period of time, as shown by step **2106**. A “correct” values is either a true value for an identity parameter or a false value for the identity parameter which is to be used in packets communicated to and from a

30 network device during the current period of time (e.g., timeslot **TS₁**, **TS₂**, **TS₃**, **TS₄**, or

TS₅ of a time frame **2020**, **2022**, **2024**, or **2026** of FIGS. 20A-20B). For example, if the current period of time is defined by timeslot TS₂ of time frame **2020**, then the “correct” value for an IP address is either TV₁ or FV₂₋₂ as specified in mission plan **1900**. In the mission plan **1900** scenario, the “correct” values for the port numbers, TCP sequence number, and MAC address are their true values. Embodiments of the present invention are not limited in this regard.

If all of the identity parameters have their “correct” values [**2108:YES**], then step **2110** is performed where the packet is forwarded to its intended destination device. Thereafter, the method **2100** returns to step **2104**. If at least one identity parameter does not have a “correct” value [**2108:NO**], then the “incorrect” value for the identity parameter is extracted from the packet, as shown by step **2112**. In a next step **2114**, the “incorrect” value is compared to at least one list of possible values for the identity parameter which is specified in a mission plan currently being employed by the computer network (e.g., computer network **100** of FIG. 1). For example, the “incorrect” value is compared to the lists of possible IP address values which are provided in mission plan **1900**. In the mission plan **1900** scenario, the comparison operations of step **2114** are performed for purposes of determining whether the “incorrect” value is an “empty” IDPV or a “shadow” IDPV associated with an IP address for a network node ND₁ or ND₂. Embodiments of the present invention are not limited in this regard.

If the “incorrect” value is not an “empty” IDPV or a “shadow” IDPV for a network node [**2116:NO**], then the packet is dropped and the method **2100** returns to step **2104**, as shown by step **2118**. If the “incorrect” value is an “empty” IDPV or a “shadow” IDPV for a network node [**2116:YES**], then the method **1200** continues with step **2118**. In step **2118**, one or more shadow networking activities is performed by the MTT enabled device as specified by a mission plan (e.g., mission plan **1900** or **1902** of FIG. 19). Such shadow networking activities include, but are not limited to, dropping the packet, forwarding the packet to an intrusion analysis tool and/or a honeypot analysis tool, logging information associated with the packet, determining the source of the packet, and/or sending a response to the source of the packet.

Intrusion analysis tools and honeypot analysis tools are well known in the art, and therefore will not be described herein. Any such known or to be known intrusion or honeypot analysis tool can be used with the present invention without limitation. The shadow networking activity performed in step **2118** is specified in the mission plan
5 currently being used by the computer network. The type of shadow networking activity performed in step **2118** can be selected based on whether the “incorrect” value is an “empty” IDPV or a “shadow” IDPV. In this regard, it should be understood that different shadow networking activities can be performed in step **2118** when the “incorrect” value is an “empty” IDPV as opposed to a “shadow” IDPV, and
10 vice versa. Upon completing step **2118**, step **2120** is performed where the method **2100** ends or other processing is performed.

Although the invention has been illustrated and described with respect to one or more implementations, equivalent alterations and modifications will occur to others skilled in the art upon the reading and understanding of this specification and
15 the annexed drawings. In addition, while a particular feature of the invention may have been disclosed with respect to only one of several implementations, such feature may be combined with one or more other features of the other implementations as may be desired and advantageous for any given or particular application. Thus, the breadth and scope of the present invention should not be limited by any of the above
20 described embodiments. Rather, the scope of the invention should be defined in accordance with the following claims and their equivalents.

CLAIMS

1. A method for identifying malicious attacks being waged on a computer network, comprising:
 - 5 implementing a mission plan at a first network node, said mission plan specifying (a) that at least one first identity parameter for a second network node has a plurality of possible values associated therewith, and (b) that at least two possible values of said plurality of possible values are to be used in communications to and from said second network node in different timeslots of a plurality of timeslots of a
10 time frame;
 - comparing, by said first network node, a value for said first identity parameter which is contained in a received packet with said plurality of possible values specified in said mission plan to determine if said value is a correct value for a current timeslot; and
 - 15 performing, by said first network node, at least one action to identify, deter or delay a possible malicious attack on said computer network if it is determined that said value is not correct for said current timeslot.
2. The method according to claim 1, wherein said identity parameter comprises a
20 port number, a TCP sequence number, an IP address, a network prefix, a host number, or a MAC address.
3. The method according to claim 1, wherein said plurality of possible values
25 comprise a true value specifying true information about said identity parameter and at least one false value specifying false information about said identity parameter.
4. The method according to claim 1, wherein said correct value comprises one of
said two possible values which is to be used in communications to and from said
second network node in said current timeslot, as specified by said mission plan.

30

5. The method according to claim 1, wherein said action comprises dropping said packet, forwarding said packet to an intrusion analysis tool or a honeypot analysis tool, logging information associated with said packet, determining a source of said packet, and/or sending a response to said source of said packet.

5

6. The method according to claim 1, wherein said action is specified in said mission plan.

7. The method according to claim 1, further comprising forwarding, by said first network node, said packet to its intended destination device if it is determined that said value is correct for said current timeslot.

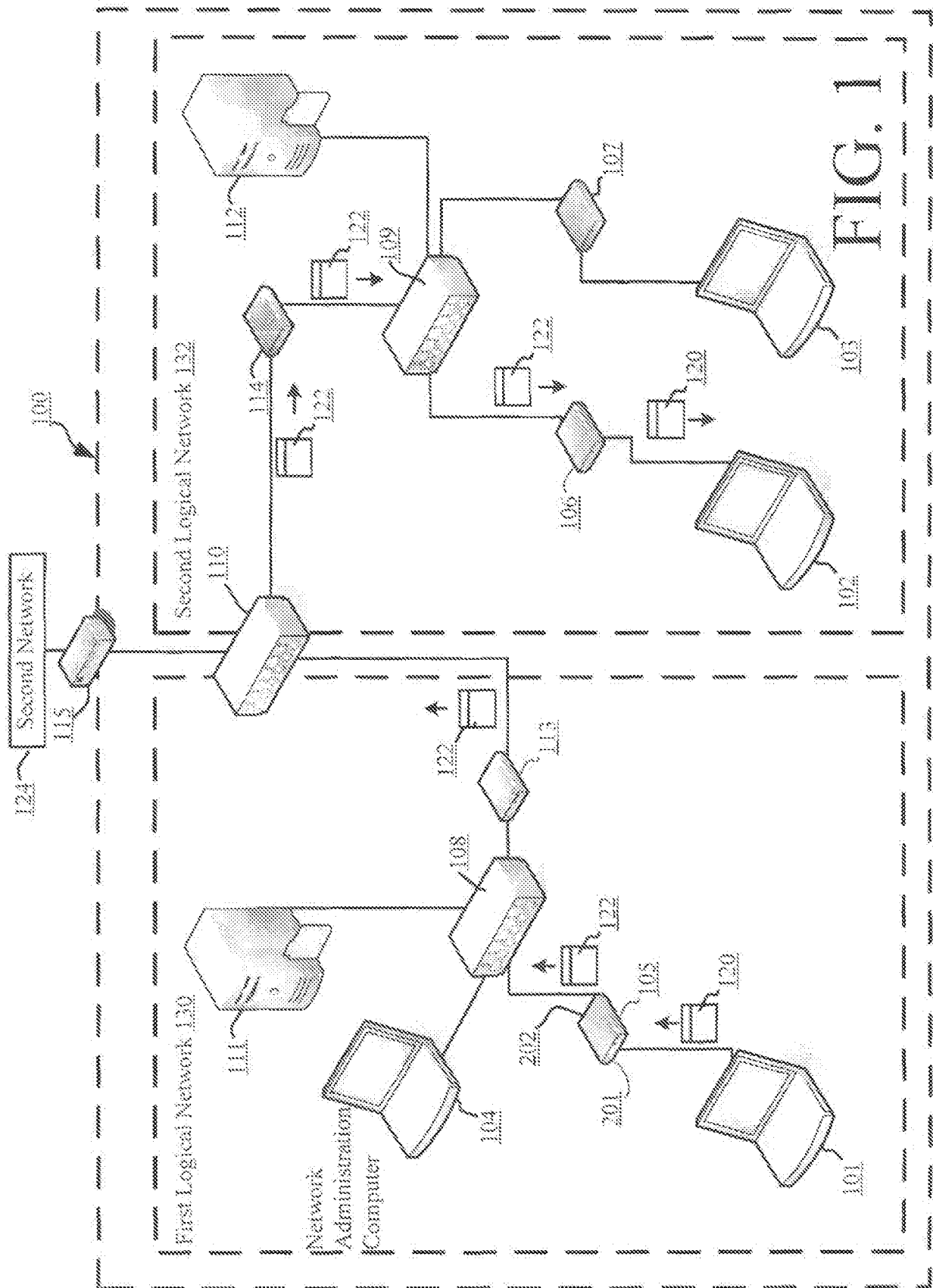
8. The method according to claim 1, wherein said value is not correct for said current timeslot when said mission plan indicates that said value is not to be used in packets communicated to and from said second network node during any of said plurality of timeslots of said time frame.

9. The method according to claim 1, wherein said value is not correct for said current timeslot when said mission plan indicates that said value was used in a packet communicated to or from said second network node during a previous timeslot of said plurality of timeslots.

10. A system, comprising:
at least on network node implementing a mission plan specifying (a) that at least one first identity parameter for a second network node has a plurality of possible values associated therewith, and (b) that at least two possible values of said plurality of possible values are to be used in communications to and from said second network node in different timeslots of a plurality of timeslots of a time frame,
where said network node is configured to

compare a value for said first identity parameter which is contained in a received packet with said plurality of possible values specified in said mission plan to determine if said value is a correct value for a current timeslot, and

- 5 perform at least one action to identify, deter or delay a possible malicious attack on said computer network if it is determined that said value is not correct for said current timeslot.



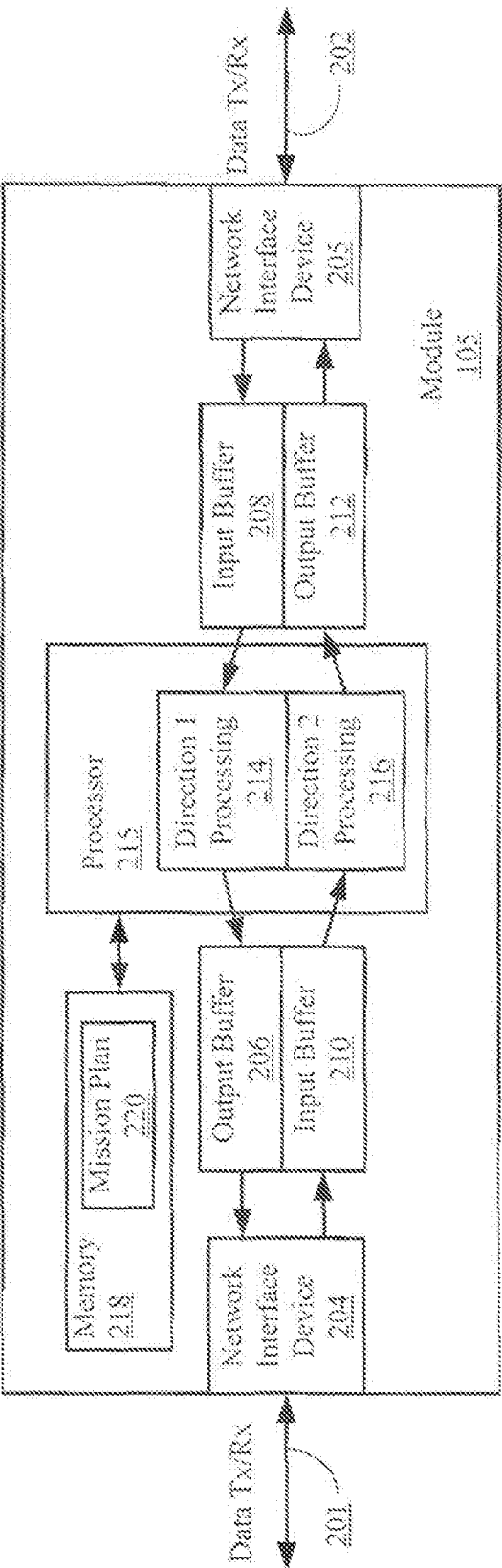


FIG. 2

3/22

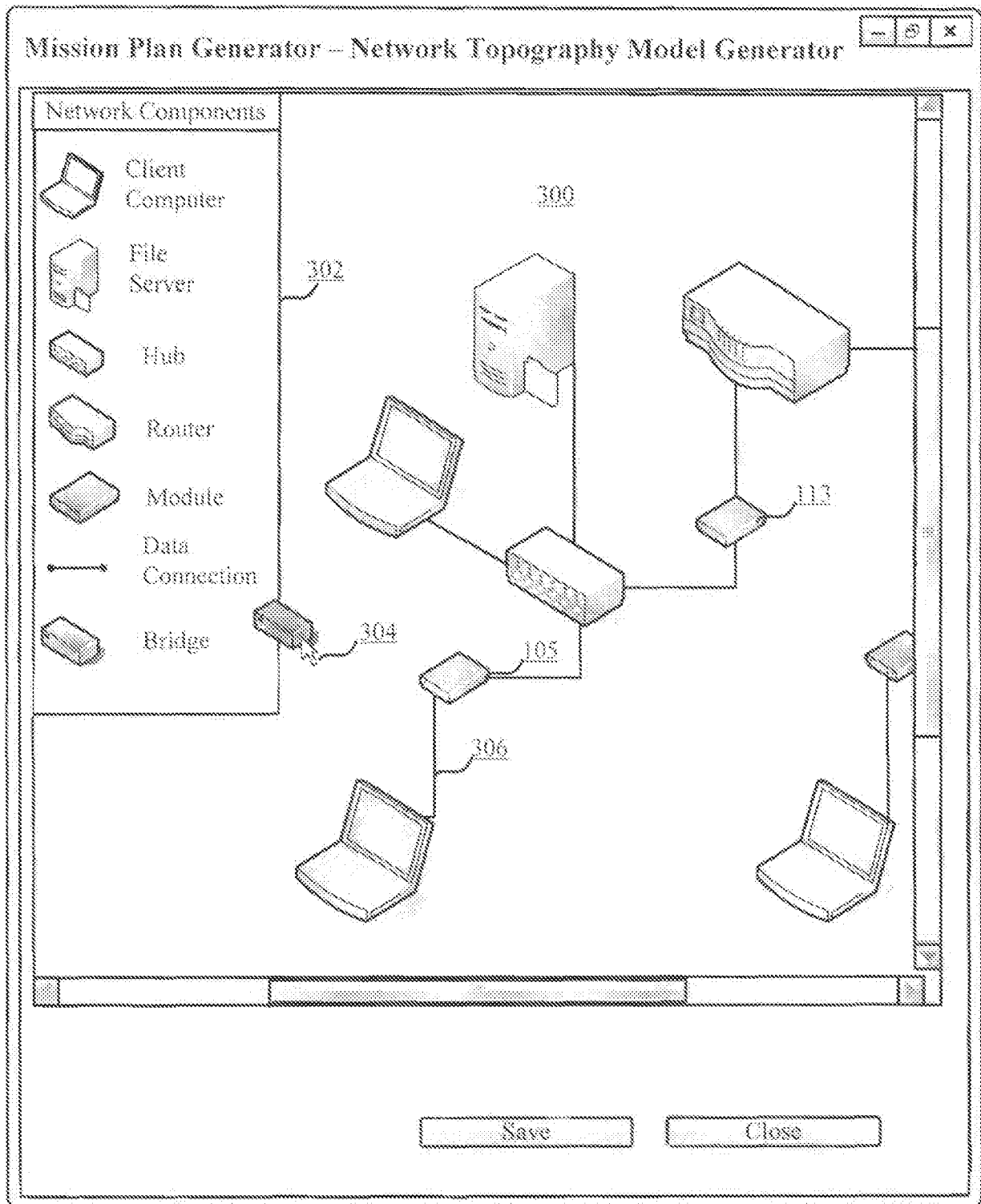


FIG. 3

4/22

Mission Plan Generator -- Dynamic Settings for Modules

401 ☒ Enable Dynamic Selection of Identity Parameters to be Varied

402 IP Address 404 MAC Address 406 Port

Module Select 432

All
Module 105
Module 106 400

408 ☒ Enable IP Address Hopping

410 ☒ Source Address

412 ☒ Destination Address

Source Address Range 422

Destination Address Range 424

Hopping Process

414 ☒ Pseudorandom Process 1

415 ☐ Pseudorandom Process 2

Trigger Type

416 ☒ Time Based

418 ☒ Periodic

☐ Aperiodic

428 ☐ Event Based

6 /hr 420 120/hr

Frequency

Select Event Based Trigger Events

430 ☐ Event type 1

☐ Event type 2

☐ Event type 3

426 Save Close

FIG. 4

5/22

500

Select the Number of Time Slots Included in a Timing Epoch

4

502

508

Define Active or Bypass Status for Each Module

Module 105

504₁ 504₂ 504₃ 504₄

Active Bypass Active Bypass

506

Module 106

Bypass Bypass Bypass Bypass

Module 107

Bypass Bypass Bypass Bypass

503

Module 113

Bypass Active Bypass Active

Module 114

504₁ 504₂ 504₃ 504₄

Active Active Active Active

510

Save Close

FIG. 5

6/22

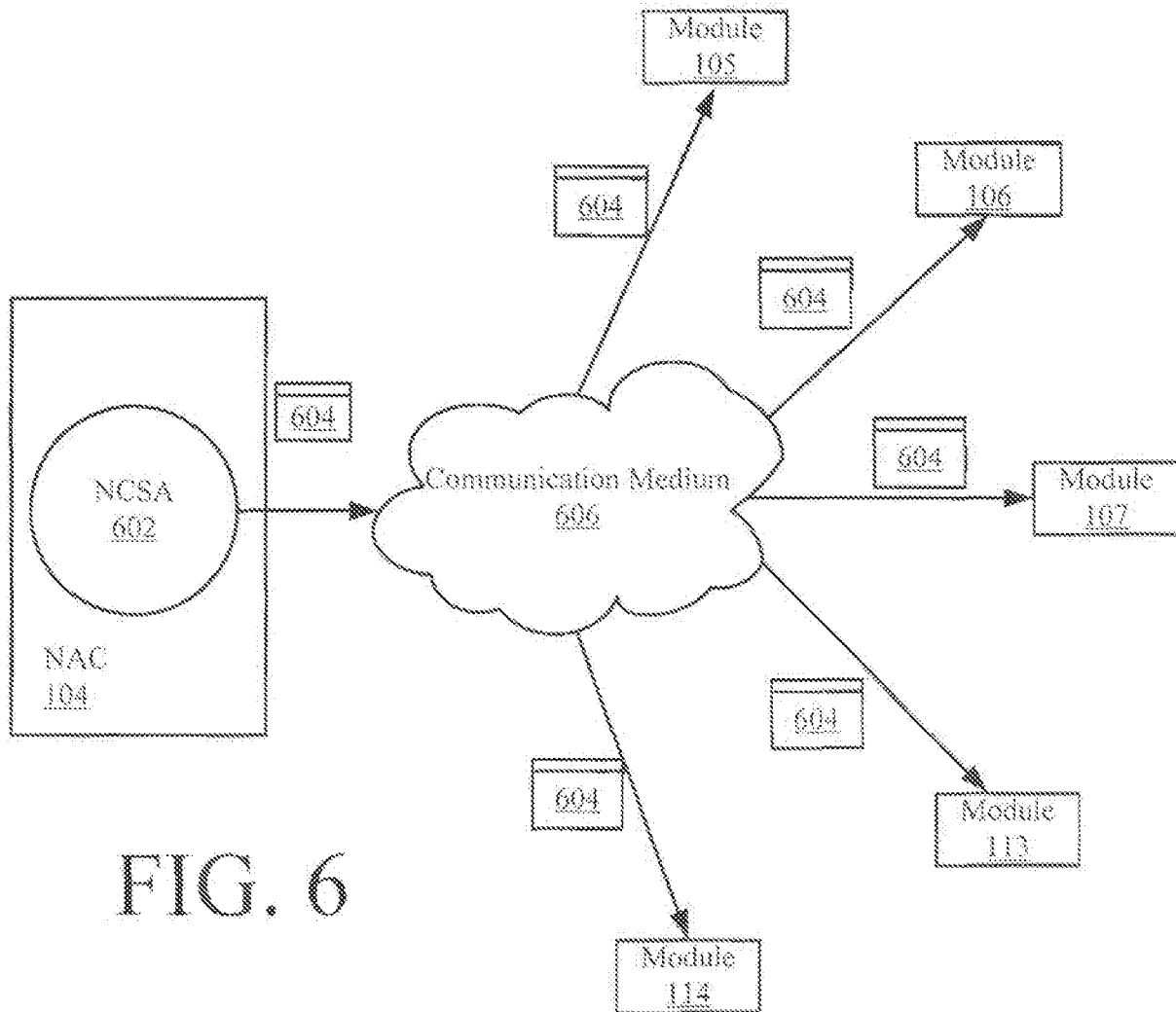


FIG. 6

7/22

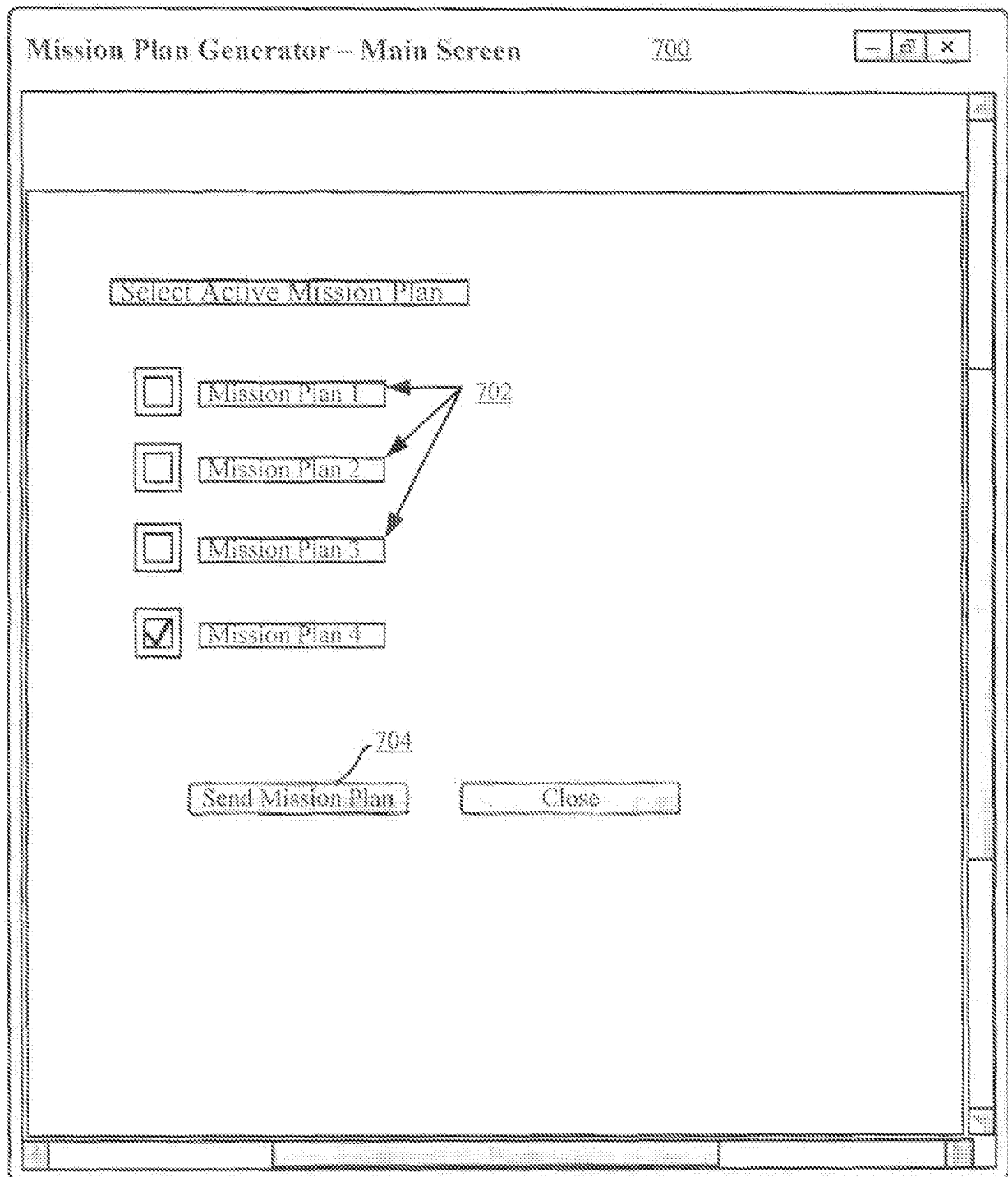
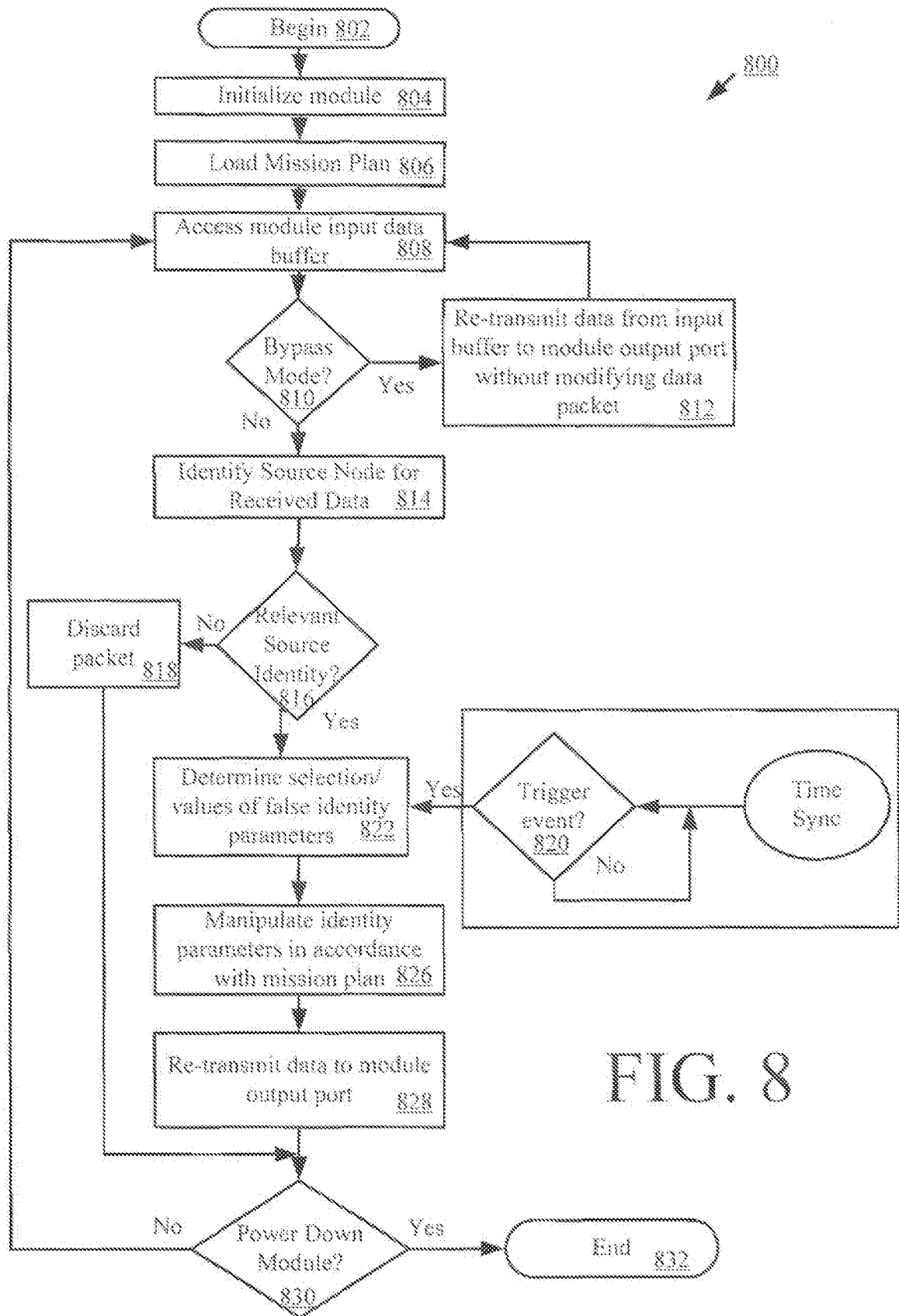


FIG. 7

8/22



9/22

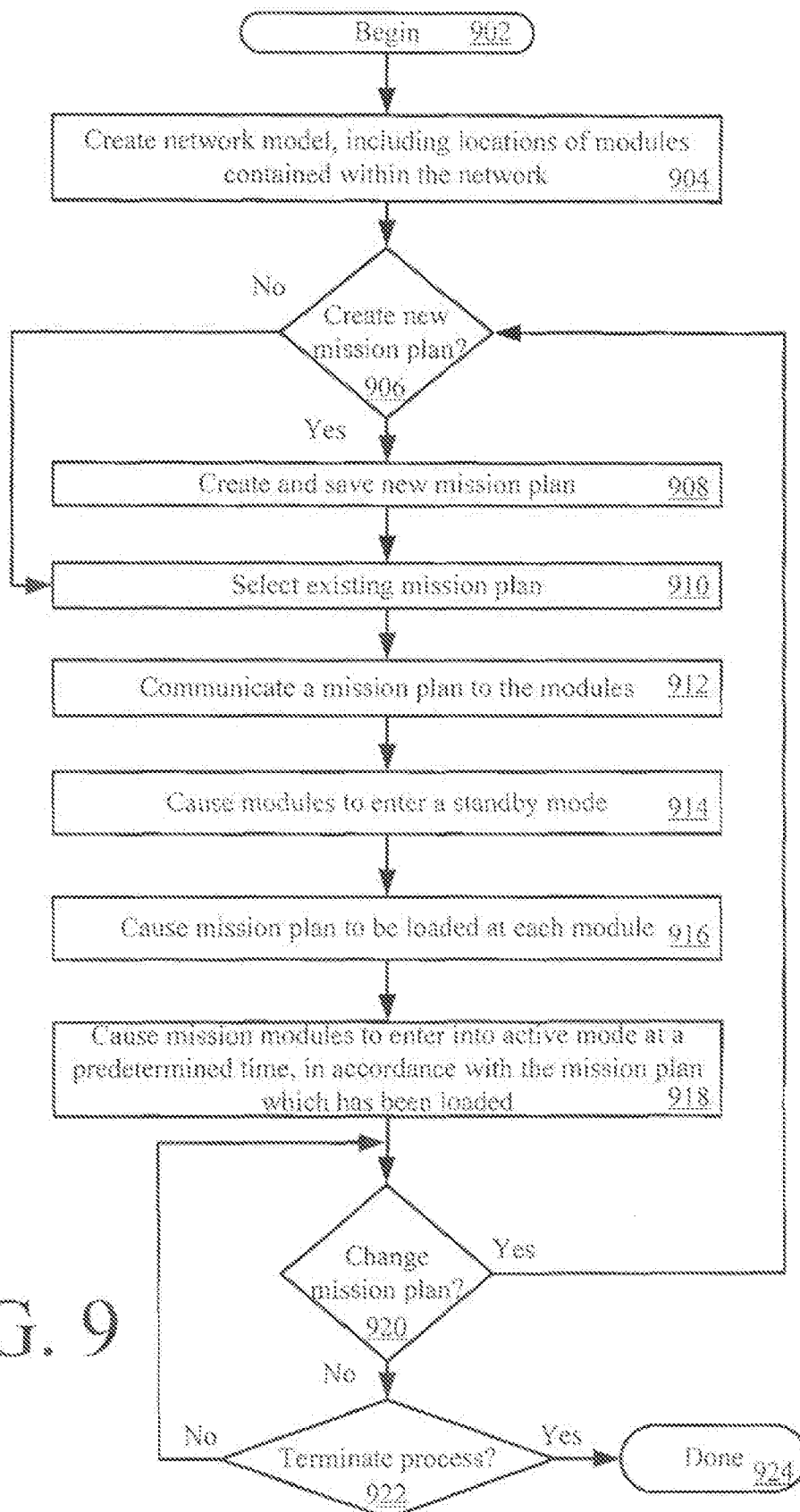


FIG. 9

10/22

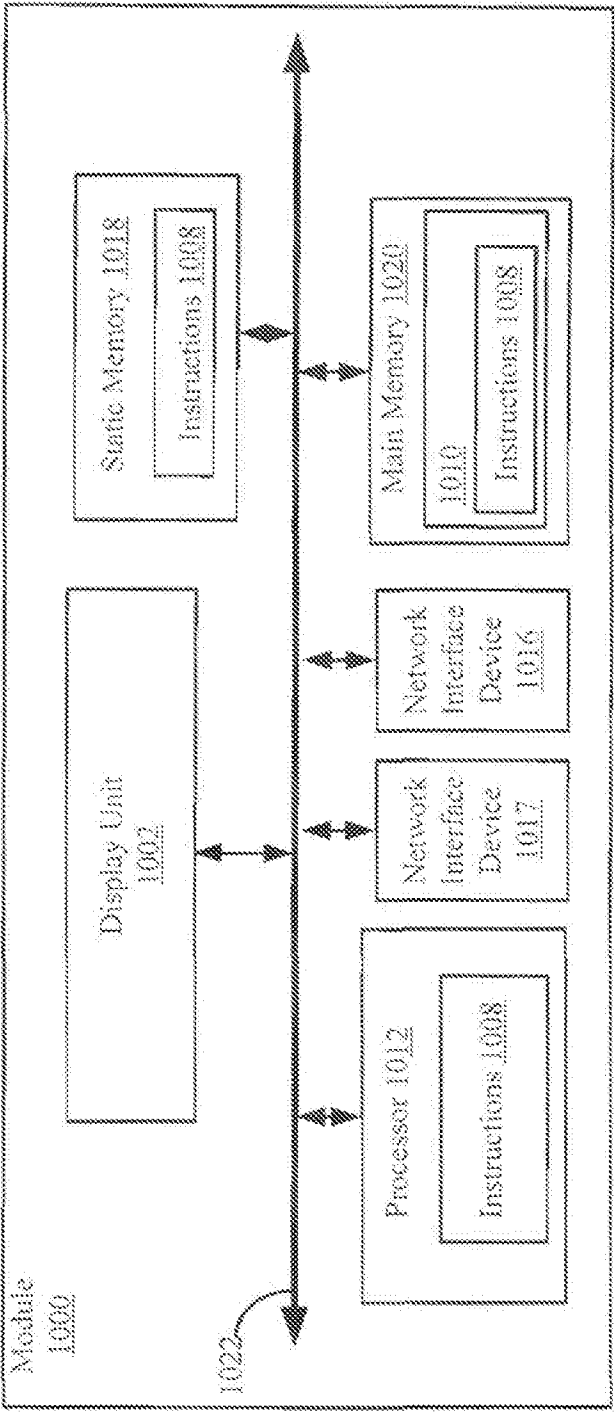


FIG. 10

11/22

Conventional Protocol Stack 1100

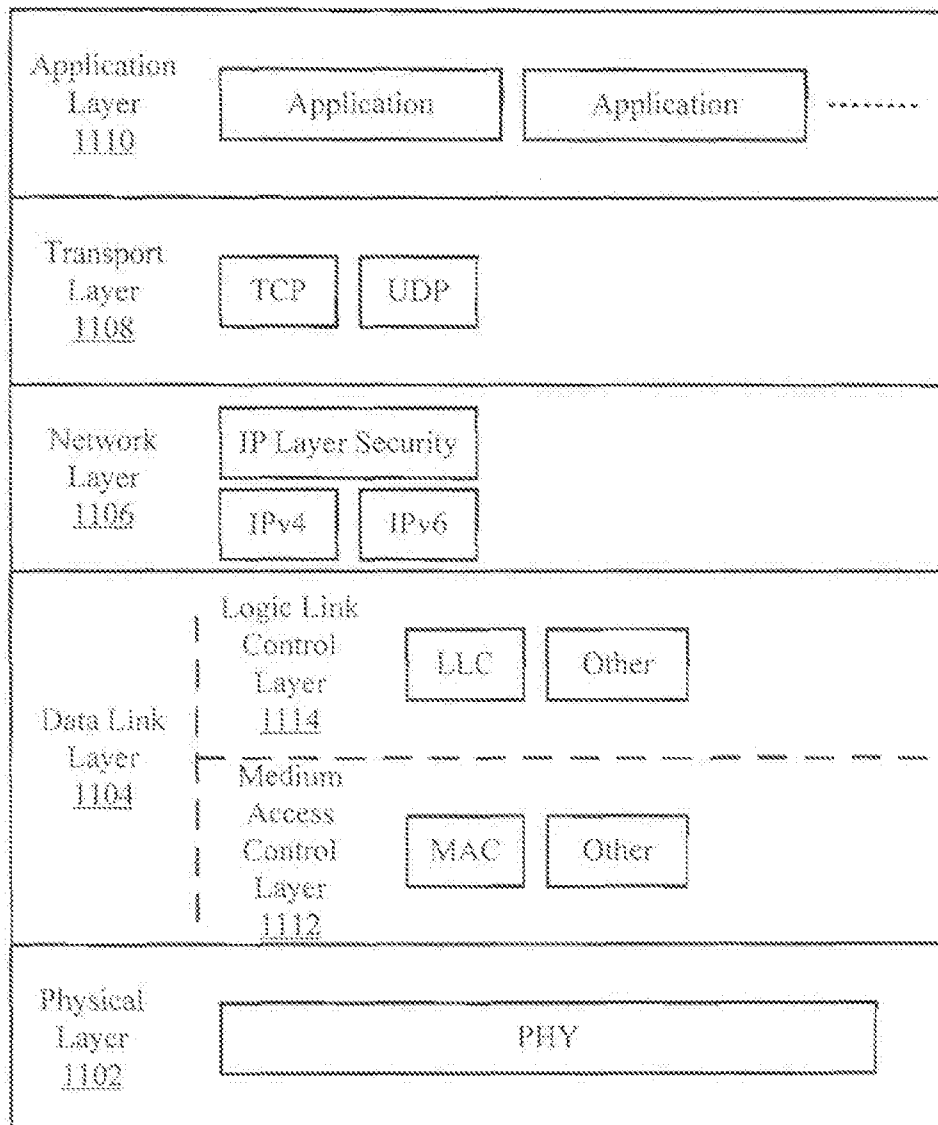


FIG. 11

(Prior Art)

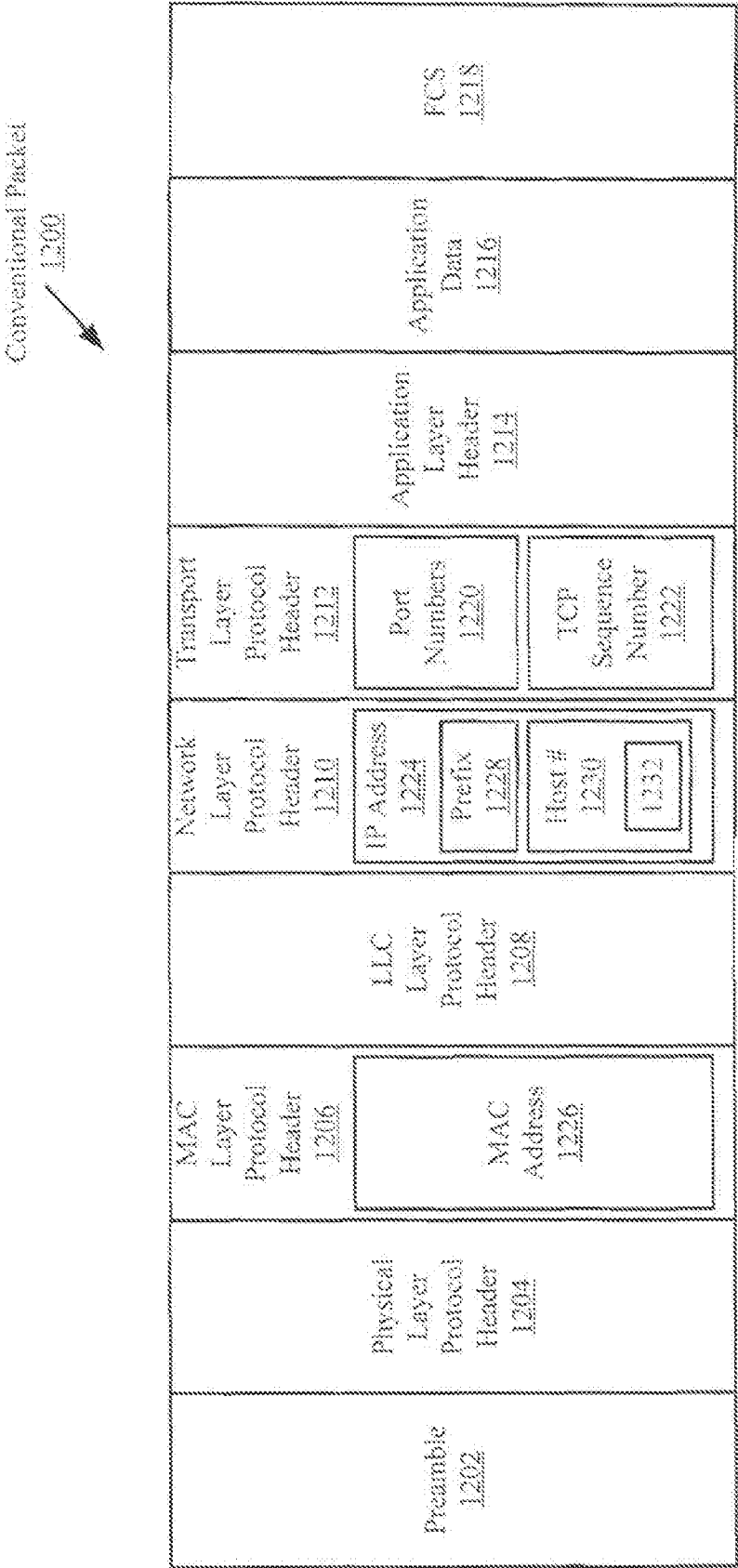


FIG. 12
(Prior Art)

13/22

MTT Protocol Stack 1300

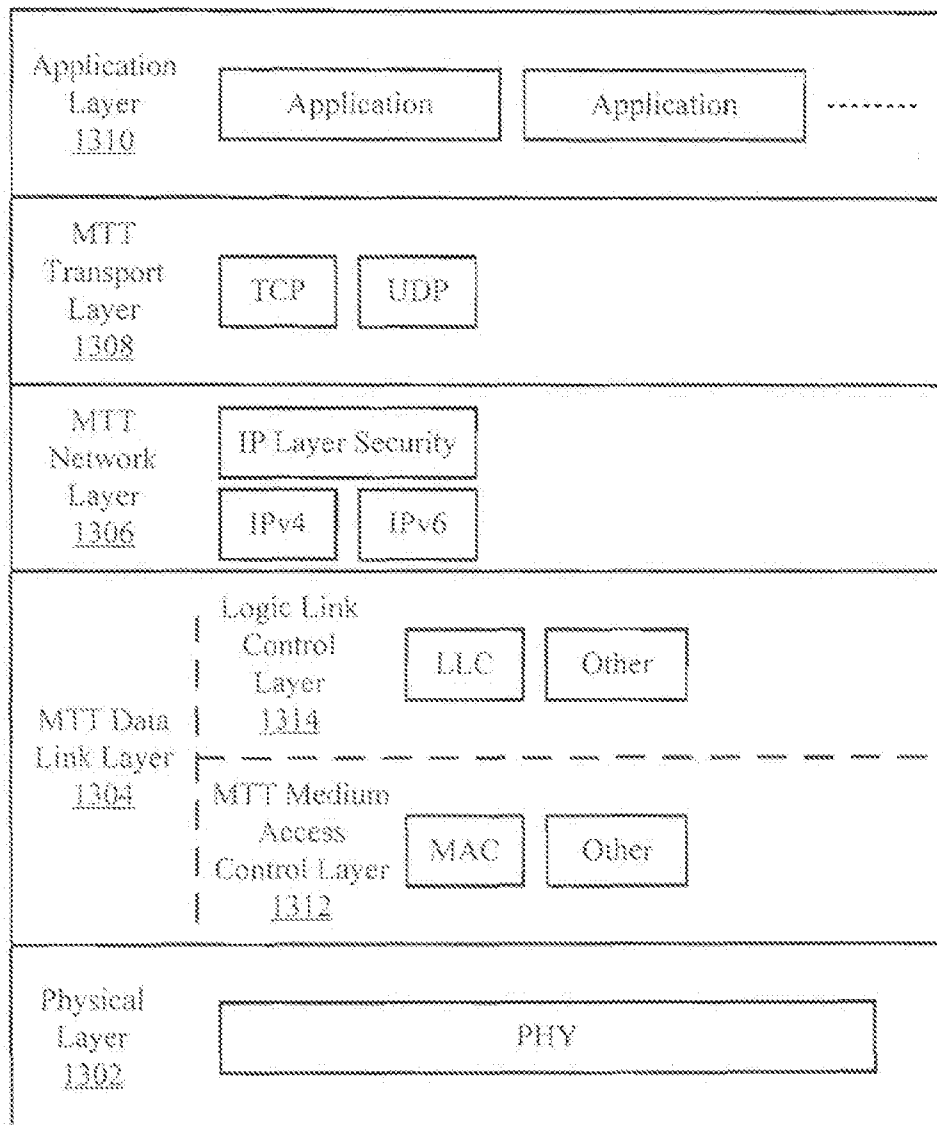


FIG. 13

14/22

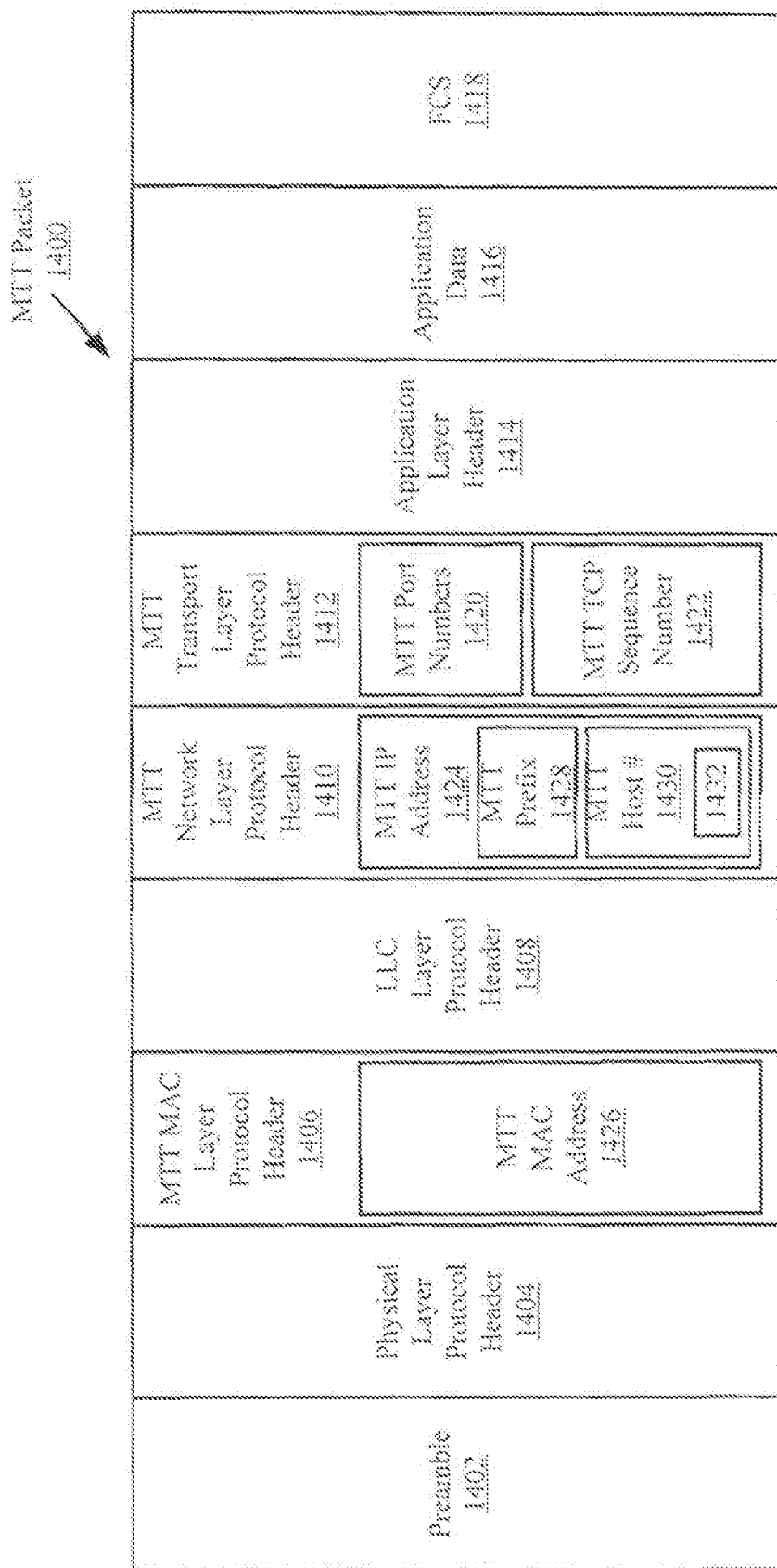


FIG. 14

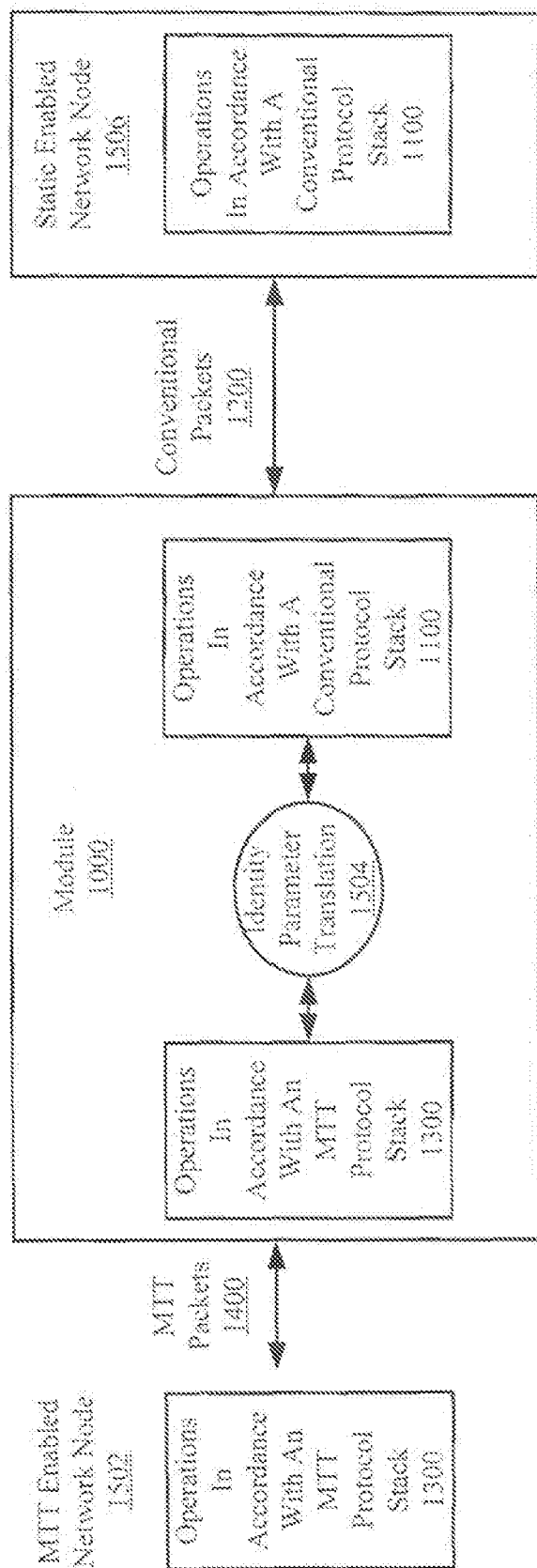


FIG. 15

16/22

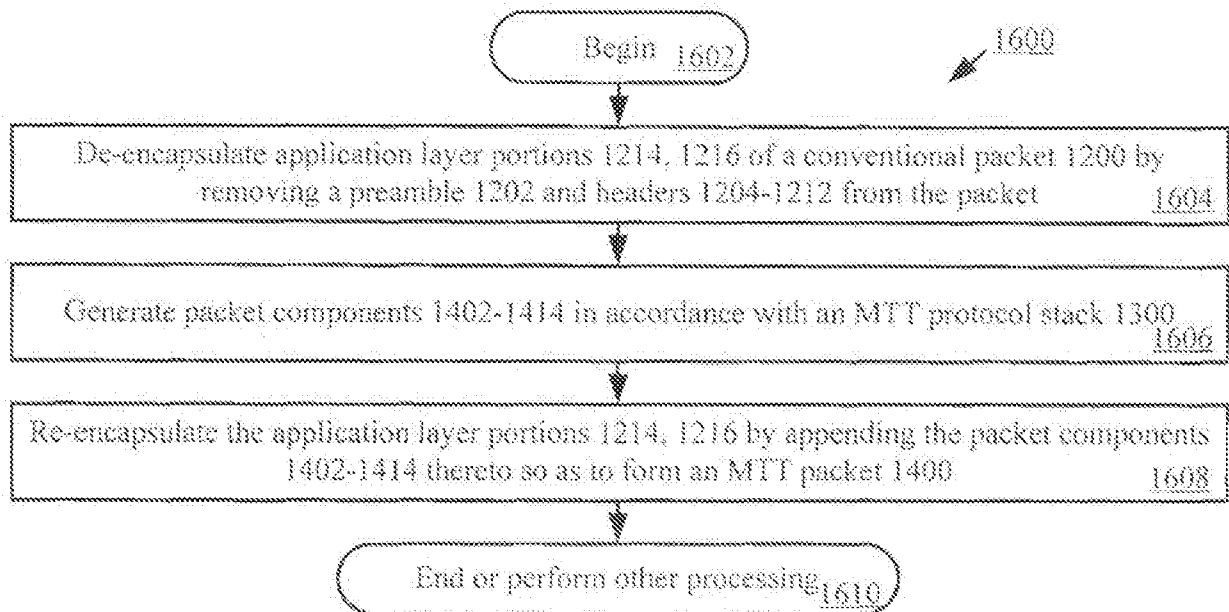


FIG. 16

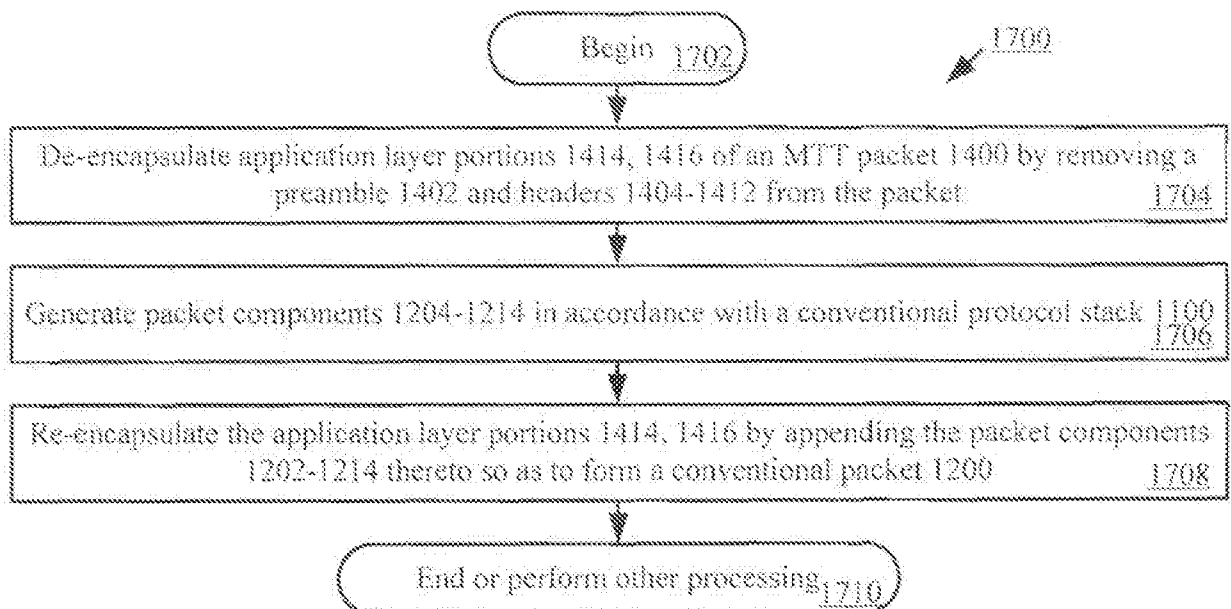


FIG. 17

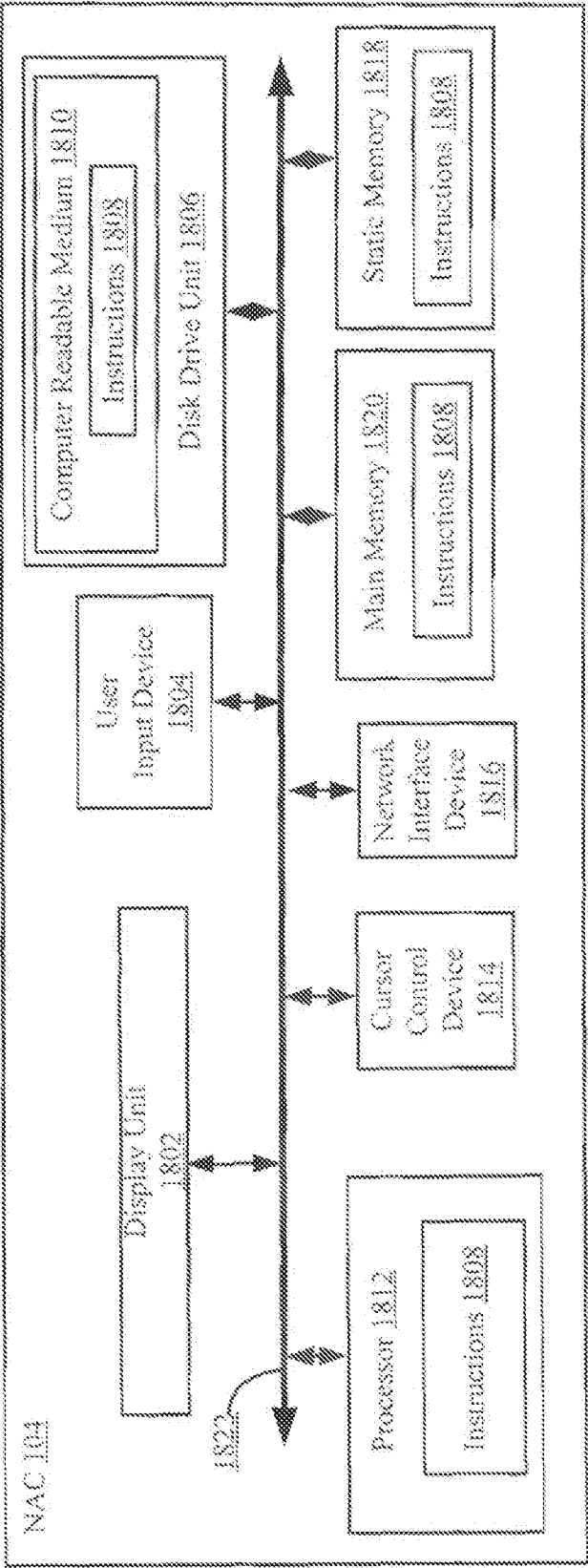


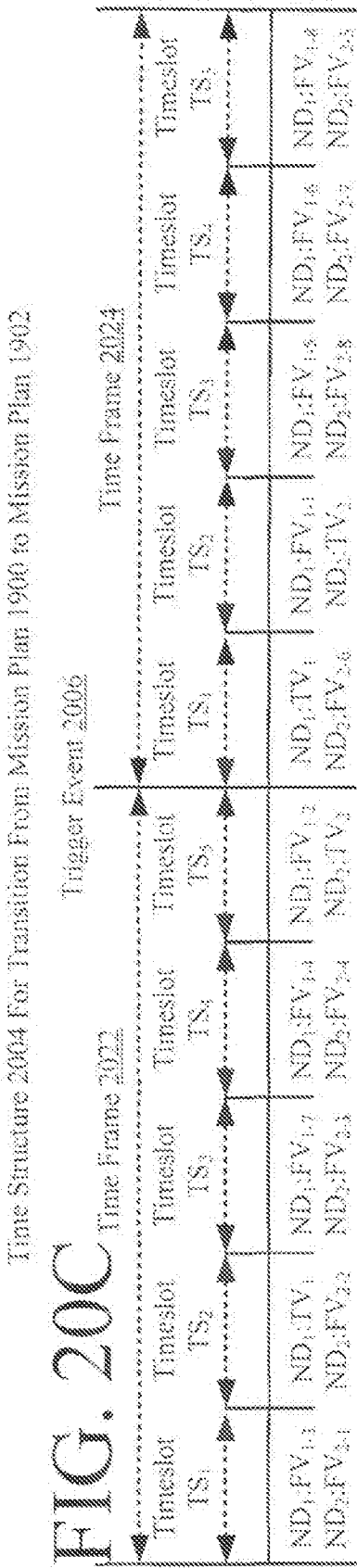
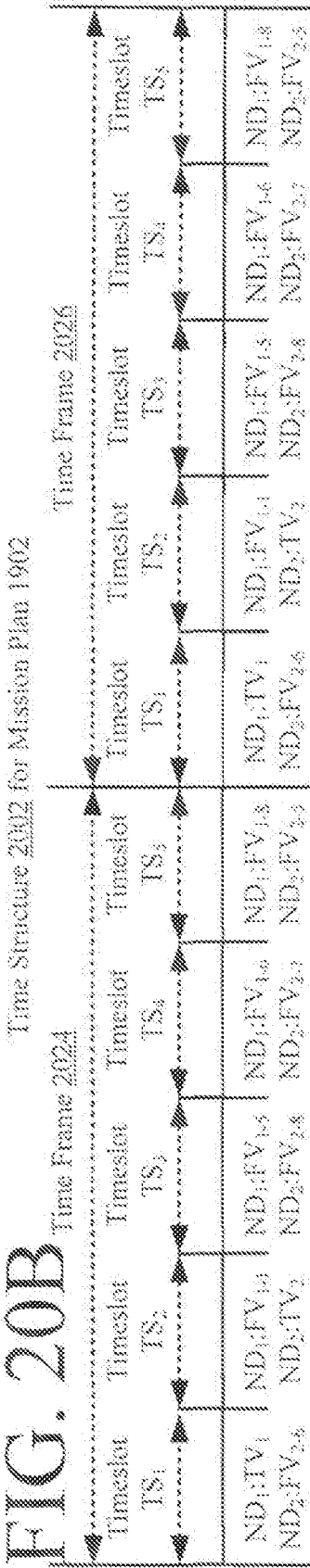
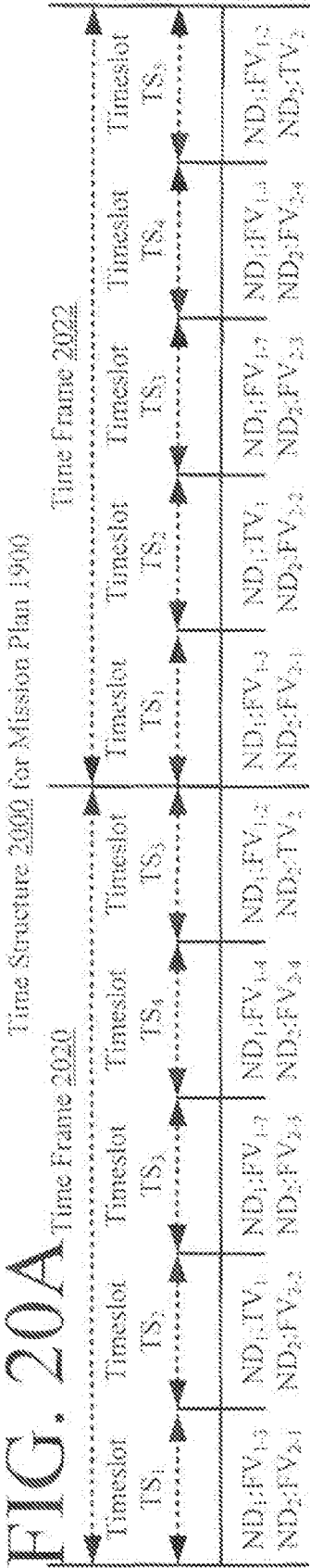
FIG. 18

18/22

Mission Plan 1900	
Variable Identity Parameter:	Mission Model 1904
(1) IP Address	
- Network Device ND ₁	
True Value TV ₁ = 175.18.252.1	Active, Timeslot TS ₁
False Value FV _{1,1} = 176.18.252.1	Empty
False Value FV _{1,2} = 175.18.252.2	Active, Timeslot TS ₂
False Value FV _{1,3} = 176.18.253.1	Active, Timeslot TS ₃
False Value FV _{1,4} = 177.18.252.2	Active, Timeslot TS ₄
False Value FV _{1,5} = 178.18.254.1	Active, Timeslot TS ₅
False Value FV _{1,6} = 179.20.252.3	Empty
False Value FV _{1,7} = 177.18.262.5	Active, Timeslot TS ₆
False Value FV _{1,8} = 179.12.263.7	Empty
- Network Device ND ₂	
True Value TV ₂ = 172.14.236.7	Active, Timeslot TS ₇
False Value FV _{2,1} = 176.18.252.1	Empty
False Value FV _{2,2} = 178.18.254.1	Empty
False Value FV _{2,3} = 179.20.252.3	Empty
False Value FV _{2,4} = 179.12.263.7	Empty
False Value FV _{2,5} = 172.16.242.9	Active, Timeslot TS ₈
False Value FV _{2,6} = 170.18.251.3	Active, Timeslot TS ₉
False Value FV _{2,7} = 170.20.242.1	Active, Timeslot TS ₁₀
False Value FV _{2,8} = 172.14.262.1	Active, Timeslot TS ₁₁
Shadow Networking Activities:	
Empty: Drop Packet, Forward To IA - Log Packet Activity	
Old: Forward To IA - Log Packet Activity, Send Response	

Mission Plan 1902	
Variable Identity Parameter(s):	Mission Model 1906
(1) IP Address	
- Network Device ND ₁	
True Value TV ₁ = 175.18.252.1	Active, Timeslot TS ₁
False Value FV _{1,1} = 176.18.252.1	Active, Timeslot TS ₂
False Value FV _{1,2} = 175.18.252.2	Empty
False Value FV _{1,3} = 176.18.253.1	Empty
False Value FV _{1,4} = 177.18.252.2	Empty
False Value FV _{1,5} = 178.18.254.1	Active, Timeslot TS ₃
False Value FV _{1,6} = 179.20.252.3	Active, Timeslot TS ₄
False Value FV _{1,7} = 177.18.262.5	Empty
False Value FV _{1,8} = 179.12.263.7	Active, Timeslot TS ₅
- Network Device ND ₂	
True Value TV ₂ = 172.14.236.7	Active, Timeslot TS ₆
False Value FV _{2,1} = 176.18.252.1	Empty
False Value FV _{2,2} = 178.18.254.1	Empty
False Value FV _{2,3} = 179.20.252.3	Empty
False Value FV _{2,4} = 179.12.263.7	Empty
False Value FV _{2,5} = 172.16.242.9	Active, Timeslot TS ₇
False Value FV _{2,6} = 170.18.251.3	Active, Timeslot TS ₈
False Value FV _{2,7} = 170.20.242.1	Active, Timeslot TS ₉
False Value FV _{2,8} = 172.14.262.1	Active, Timeslot TS ₁₀
Shadow Networking Activities:	
Empty: Forward To IA - Log Packet Activity, Send Response	
Old: Forward To IA - Log Packet Activity, Send Response	

FIG. 19



20/22

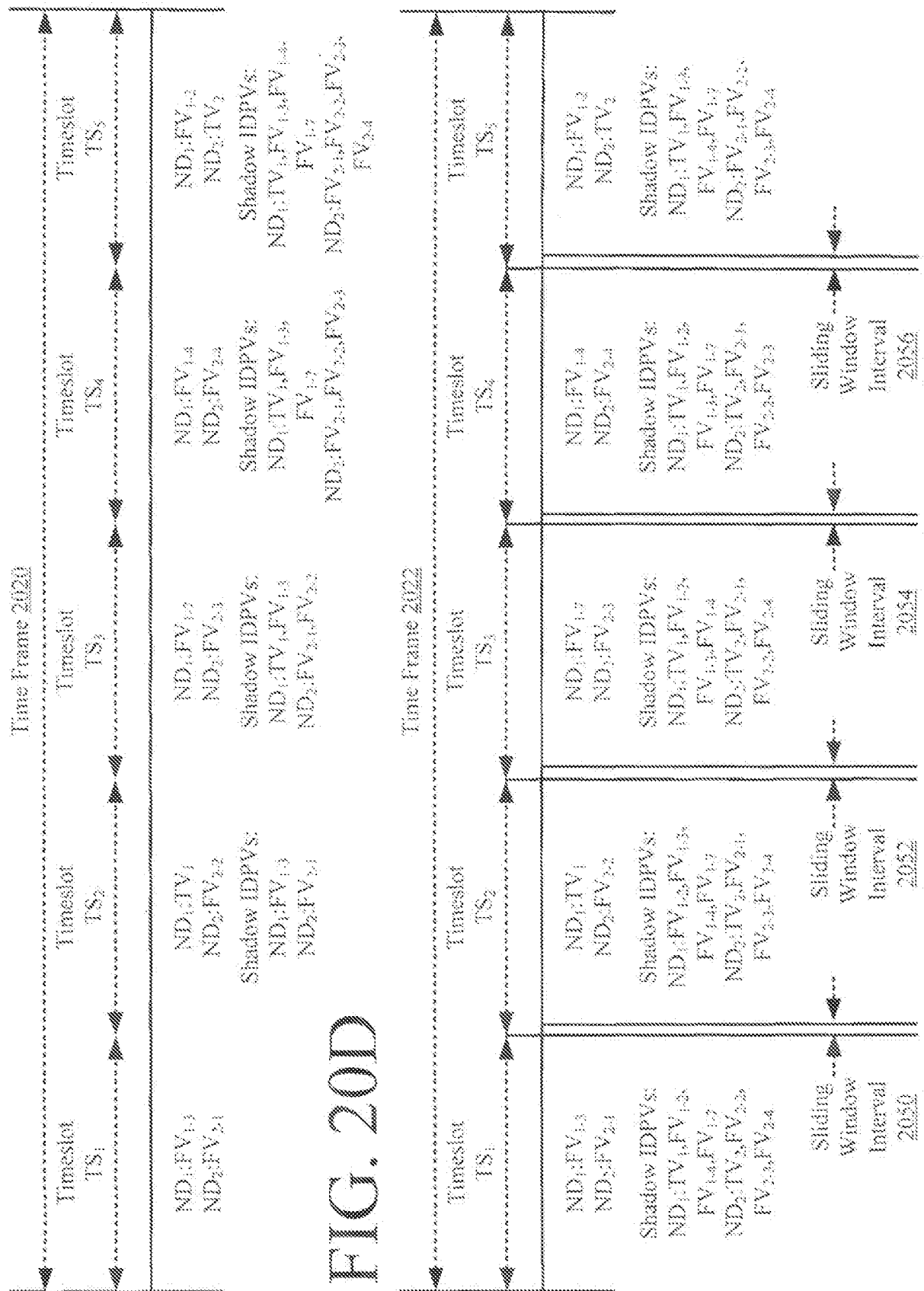
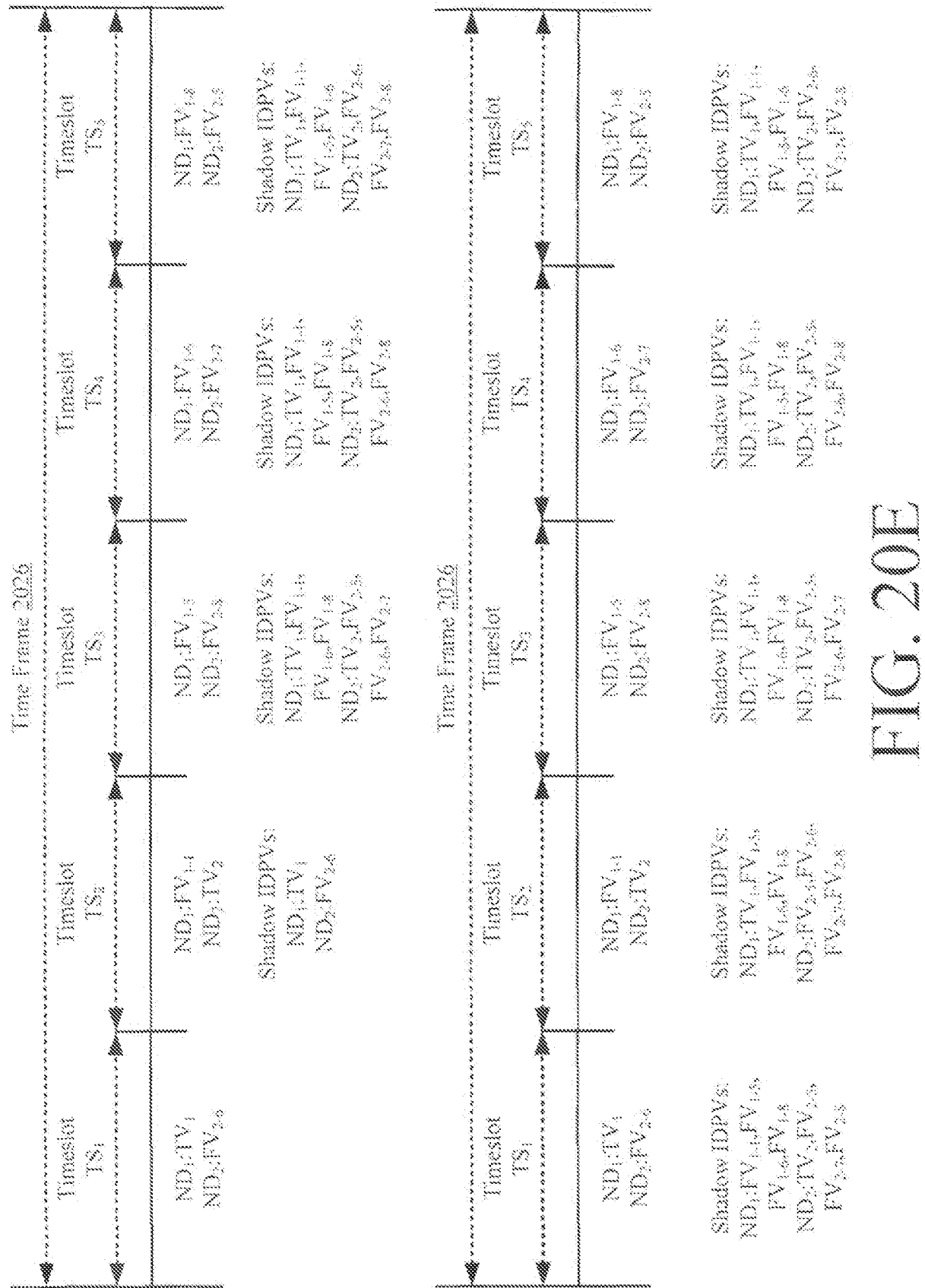


FIG. 20D



22/22

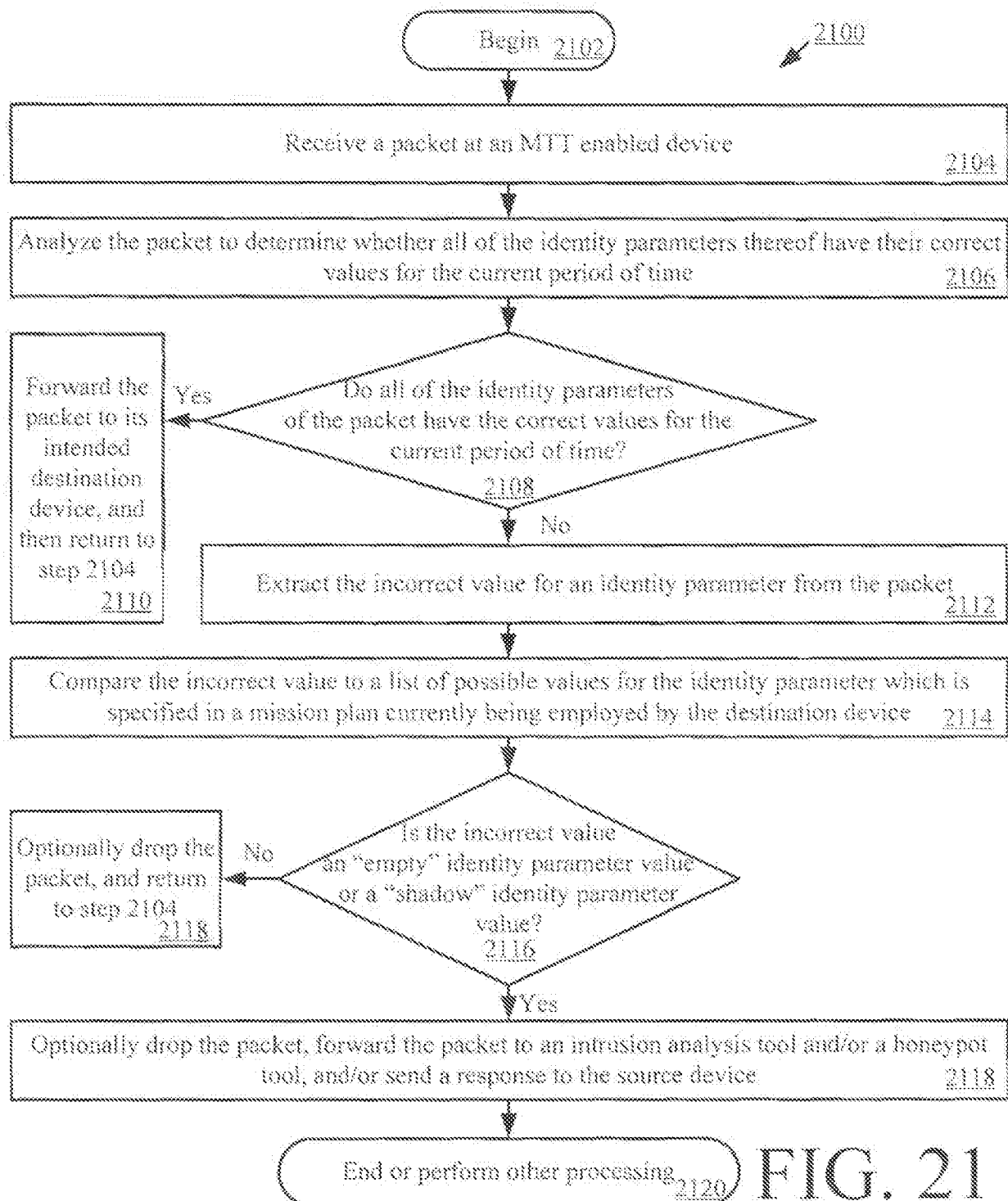


FIG. 21