



(51) International Patent Classification:

H04L 41/147 (2022.01) H04L 43/0852 (2022.01)
H04L 41/16 (2022.01) H04L 41/122 (2022.01)
H04L 41/0823 (2022.01) H04L 41/149 (2022.01)

(21) International Application Number:

PCT/EP2021/081460

(22) International Filing Date:

12 November 2021 (12.11.2021)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

20306390.4 16 November 2020 (16.11.2020) EP

(71) Applicant: **INTERDIGITAL CE PATENT HOLDINGS, SAS** [FR/FR]; 3 rue du Colonel Moll, 75017 Paris (FR).

(72) Inventors: **BONNIOT, Loick**; 44 rue de Rennes, 35510 Cesson-Sevigne (FR). **NEUMANN, Christoph**; c/o InterDigital R&D France, SAS, 975 Avenue des Champs Blancs, 35576 Cesson-Sevigne (FR). **SCHNITZLER, Francois**; c/o InterDigital R&D France, SAS, 975 Avenue des Champs Blancs, 35576 Cesson-Sevigne (FR). **TAÏANI, François**; Université de Rennes 1, 263 Avenue du Général Leclerc - Bât 12, Campus universitaire de Beaulieu, 35042 RENNES Cedex (FR).

(74) Agent: **INTERDIGITAL**; 975, avenue des Champs Blancs, 35576 CESSON-SEVIGNE (FR).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, 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, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, ST, 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, KM, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

(54) Title: METHOD AND SYSTEM FOR PREDICTING A NETWORK CHARACTERISTIC

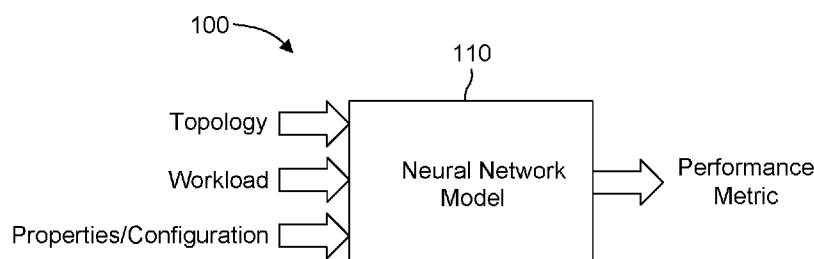


FIG. 1

(57) Abstract: A graph neural network with three bipartite graphs based on a topology of a network, $G_{p,l}$ connecting paths in a network to their links, $G_{l,n}$ connecting links to nodes defining the links, and $G_{p,n}$ connecting paths to nodes, wherein each vertex of the bipartite graphs has a hidden state to be passed to its neighbours during iterated message passing convolutions, the graph neural network is injected network parameters and load are injected into the above graphs. A network characteristic combines the resulting outputs using a fully connected layer.



METHOD AND SYSTEM FOR PREDICTING A NETWORK CHARACTERISTIC

TECHNICAL FIELD

The present disclosure relates generally to networks and in particular to network analysis.

5 BACKGROUND

This section is intended to introduce the reader to various aspects of art, which may be related to various aspects of the present disclosure that are described and/or claimed below. This discussion is believed to be helpful in providing the reader with background information to facilitate a better understanding of the various aspects of the present disclosure. Accordingly, it should be understood that these statements
10 are to be read in this light, and not as admissions of prior art.

Networks, such as communication networks or electricity networks, can be complex distributed systems for which every configuration parameter (e.g. network routes, priority policies, and offloading policies) and the topology of the network itself
15 can have an important impact on the network's performance. For example, in a communication network the routing policy and the priority policies will largely affect the resulting latency of traffic flows. Similarly, in an electricity network, decisions to route electricity into certain parts of the network have a large impact on the overall capacity of the network. It is therefore typically a complicated task to take adequate
20 and informed decisions (e.g. where to route a traffic flow or where to produce electricity), in particular if the network topology evolves (e.g. a new node or link is added or removed due to failure or decision). In order to help in this context, machine learning models could be used to either predict performance for a given network and configuration, or to evaluate what-if scenarios (e.g. what is the performance if a link
25 is removed or if the routing configuration is X or Y).

It will thus be appreciated that there is a desire for a solution for network analysis. The present principles provide such a solution.

SUMMARY OF DISCLOSURE

In a first aspect, the present principles are directed to a method comprising
30 obtaining a network model for predicting a characteristic of a network, the network

model comprising three bipartite graphs corresponding to a topology of the network, wherein a first bipartite graph connects each path in the network to links the path uses in the network, a second bipartite graph connects each link in the network to nodes that define the link, and a third bipartite graph connects each path in the network to the nodes that define the path, wherein each vertex of the bipartite graphs contains a state, injecting parameters of the network and load in the network into the network model, performing T iterations of updating the states of at least some of the vertices based on current states of the vertices and states of neighbour vertices, combining the state of every path to obtain a prediction of the characteristic, and outputting the prediction of the characteristic.

In a second aspect, the present principles are directed to an apparatus comprising memory and a processor configured to obtain a network model for predicting a characteristic of a network, the network model comprising three bipartite graphs corresponding to a topology of the network, wherein a first bipartite graph connects each path in the network to links the path uses in the network, a second bipartite graph connects each link in the network to nodes that define the link, and a third bipartite graph connects each path in the network to the nodes that define the path, wherein each vertex of the bipartite graphs contains a state, inject parameters of the network and load in the network into the network model, perform T iterations of updating the states of at least some of the vertices based on current states of the vertices and states of neighbour vertices, combine the state of every path to obtain a prediction of the characteristic, and output the prediction of the characteristic.

In a third aspect, the present principles are directed to a computer program product which is stored on a non-transitory computer readable medium and includes program code instructions executable by a processor for implementing the steps of a method according to any embodiment of the first aspect.

BRIEF DESCRIPTION OF DRAWINGS

Features of the present principles will now be described, by way of non-limiting example, with reference to the accompanying drawings, in which:

FIG. 1 illustrates the present principles generally;

FIG. 2 illustrates an example of input network topology for use with the present principles;

FIG. 3 illustrates a bipartite graph $G_{p,l}$ from paths to links for the input network topology illustrated in FIG. 2;

5 FIG. 4 illustrates the bipartite graph $G_{l,n}$ from links to nodes for the input network topology illustrated in FIG. 2;

FIG. 5 illustrates the bipartite graph $G_{p,n}$ from paths to nodes for the input network topology illustrated in FIG. 2;

FIG. 6 illustrates a typical example of the prediction that can be made and the type
10 of input that can be used for the prediction according to the present principles;

FIG. 7 illustrates an analysis system according to an embodiment of the present principles; and

FIG. 8 illustrates a method in a 5G network according to the present principles.

DESCRIPTION OF EMBODIMENTS

15 It will be appreciated that it desirable to provide a solution that adequately can estimate performance metrics given a network snapshot (e.g. a network topology, a network configuration which includes routing and priority configuration). In general terms, the present principles, generally illustrated in FIG. 1, should be able to predict performance metrics given: (i) a network topology, (ii) a network configuration or more
20 generally properties of the physical/target network, and (iii) a description of the workload (e.g. network or electricity flows) carried on the physical/target network. For example, in the case of a communication network, the present principles, illustrated in FIG. 6, may predict the resulting per-source-destination mean per-packet delay (or other metrics such as throughput, packet loss etc.) given: (i) a network topology, (ii)
25 a network configuration (routing, queue scheduling), and (iii) a source-destination traffic matrix.

Generally speaking, the present principles use a graph neural network that relies on three bipartite graphs, $G_{p,l}$, $G_{l,n}$, $G_{p,n}$, where paths (a path is defined by a source and destination node and the path traverses a series of nodes), nodes and
30 links of the physical network are represented as vertices:

- $G_{p,l}$ connects each path to the links it is made up of,
- $G_{l,n}$ connects each link to the two nodes that define the link, and

- $G_{p,n}$ connects each path to the nodes it traverses.

Each vertex of the above graphs has a hidden state (also called ‘embedding,’ the terms can be used interchangeably). As will be described, the hidden state is learned during a training phase using message passing convolutions between
5 vertices.

Message passing convolutions between the vertices is used during inference: the current network topology (real, i.e. physical, or modelled) defines the topology of the graphs $G_{p,l}$, $G_{l,n}$, $G_{p,n}$; the network parameters and load are injected into the above graphs and each vertex updates the parameters using its hidden state and passes
10 the result to its neighbours (each neighbour aggregating the received messages). The final prediction combines the resulting outputs using a fully connected layer.

As will also be described, the present principles can be applied to communication networks using Software Defined Networking (SDN), in particular to predict per-source-destination mean per-packet delay.

15 As will also be described, the present principles can be applied to 3GPP communication networks, in particular to predict network performance within the Network Data Analytics Function (NWDAF). A Session Management Function (SMF) may reconfigure the 3GPP network (e.g. by allocating resources or reconfiguring PDU-session) according to the predicted network performance.

20 In a variant of the present principles, during inference the system can test different network topologies and network configurations to execute what-if scenarios. The system can choose the best result and configure the network accordingly.

In another variant of the present principles, each node of the physical network hosts vertices of the graphs $G_{p,l}$, $G_{l,n}$, $G_{p,n}$. During inference a node transmits on the
25 physical network a message containing the messages of the machine-learning message-passing algorithm.

Deep learning model

The present principles are based on a deep learning model capable of estimating per-path performances for arbitrary network topologies. First will be

described how to model problem inputs in a way suitable for deep learning (and more specifically graph neural networks), then the full model architecture will be detailed and finally how embeddings are aggregated.

Modelization

5 Problem inputs are modelled using three entities: nodes (vertices of the network), links (edges of the network) and finally paths (extracted from routing configuration and network topology for each source-destination pairs of nodes). It is noted that the network can be a real, physical network or a model of a network (a.k.a. “target network”), such as a future network or a network resulting from a what-if
10 scenario.

Three bipartite graphs are constructed for the network:

- $G_{p,l}$ uses paths and links as vertices.

There is one edge between path p and link l if and only if p passes through link l .

- 15 • $G_{l,n}$ uses links and nodes as vertices.

For every link, there is one edge to the node that is at one side of the link, and another edge to the node that is at the other side of the link.

- $G_{p,n}$ uses paths and nodes as vertices.

20 There is one edge between path p and node n if and only if p passes through node n .

Start and end nodes are included for each path.

FIG.2 illustrates an example of input network topology (the physical network, such as a communication network or a power grid) with nodes A, B, C and links 1 to 4. Paths from X to Y are denoted XY (in this example path BC is constructed by
25 following links 2 and 3 via A). FIGS. 3, 4 and 5 depict the three bipartite graphs constructed from the example network topology in FIG. 2: FIG. 3 illustrates the bipartite graph $G_{p,l}$ from paths to links, FIG. 4 illustrates the bipartite graph $G_{l,n}$ from links to nodes, and FIG. 5 illustrates the bipartite graph $G_{p,n}$ from paths to nodes.

For a bipartite graph, the neighbours of i , $N_{S,T}(i)$ are defined as the indices j of vertices where (j, i) is an edge in $G_{S,T}$. Conversely, $\tilde{N}_{S,T}(i)$ is defined as the indices j of vertices where (i, j) is an edge in $G_{S,T}$. For instance, the following equalities arise from FIG. 4:

- The neighbours of link 1 are nodes A and B, i.e $N_{l,n}(1) = \{A; B\}$
- The neighbours of node A are links 1 to 4, i.e $\tilde{N}_{l,n}(A) = \{1; 2; 3; 4\}$

Model architecture

Paths, links and nodes all have a hidden state (“embedding”), updated in the model by an iterative process (repeated a configurable number of times, T). From the hidden state, it is possible to extract the final per-path predictions. The general architecture of the model is described as follows:

1. Initialization

- a. p_i is defined as the initial hidden state of path i
- b. l_i is defined as the initial hidden state of link i
- c. n_i is defined as the initial hidden state of node i

2. Computation

a. Repeat T times:

- i. For every link i , update l_i as a combination of its current value and the aggregation of every p_j such that j in $N_{p,l}(i)$. This corresponds to one “message passing” where p_j is the message being passed.
- ii. For every node i , update n_i as a combination of its current value and the aggregation of every l_j such that j in $N_{l,n}(i)$. This corresponds to one “message passing” where l_j is the message being passed.
- iii. For every link i , update l_i as a combination of its current value and the aggregation of every n_j such that j in $\tilde{N}_{l,n}(i)$. This corresponds to one “message passing” where n_j is the message being passed.
- iv. For every path i , update p_i as a combination of its current value and the aggregation of every l_j such that j in $\tilde{N}_{p,l}(i)$. This

corresponds to one “message passing” where l_j is the message being passed.

b. For every path i , update p_i as a combination of this current value and the aggregation of every n_i such that j in $\tilde{N}_{p,n}(i)$

5 c. Combine every path’s hidden state p_i to the desired predictions y_i

When updating hidden states, the present principles use “aggregation” and “combination” as generic functions. Examples of “aggregation” operations suitable for deep learning are provided hereinafter in “Hidden state aggregation”. In neural networks, “combination” is typically a single fully-connected layer, but multiple fully-connected layers can be chained with non-linear activation functions between them.

In the parlance of graph neural networks, every step in 2.a and step 2.b is a convolution operation, aggregating hidden states from multiples edges in the invention’s bipartite graphs to return updated hidden states.

As in every machine learning model, a training phase is required to learn a set of functions and parameters. According to the present principles, the same architecture is used during training and inference; during training the functions “aggregation” and “combination” are learnt and require access to input and output parameters. In other words, in order to train a model, the system requires the value of the performance metric to be predicted, a network topology, a network configuration or more generally properties of the physical or target network, and a description of the workload (e.g. network or electricity flows) carried on the physical or target network

Hidden state aggregation

The present principles can be used with any suitable kind of aggregation function, as long as the said function accepts a variable number of arguments. In the literature of graph neural network convolutions, the “Global Sum” function is largely used. For instance, the aggregation operation in 2.a.i can be:

$$l_i = \sum_{j \in N_{p,l}(i)} p_j$$

Min, Max, Average and Standard deviation functions can also be used in the aggregation. It is also possible to use several of these functions and concatenate their

results. However, a drawback of these functions is that changing the order of the inputs does not change the result (i.e. information about input order is lost).

As another example, recurrent neural networks can be used as aggregation functions when the order of input conveys important information. In the present principles, paths can be defined as ordered lists of links and/or ordered lists of nodes. When aggregating links hidden states in 2.a.iv, it is possible to feed the hidden states to a recurrent neural network (e.g. containing Long Short-Term Memory, LSTM, or Gated Recurrent Unit, GRU, layers) in the order defined by the path. Taking the example of path BC in FIG. 2, this means first feeding link 2 and then link 3 to the recurrent neural network. The same logic applies to step 2.b.

Application to communication networks

Software-Defined Networks (SDN) is the concept of orchestrating and controlling communication networks using software/controller nodes. With SDN, networking tasks such as routing, policy decisions etc. are orchestrated from a centralized control plane (using e.g. a central controller), which may leverage the network information (including the network topology, the network state such as the current traffic workload and other performance metrics) in order to operate networks efficiently and dynamically adapt to changes in the network. An SDN controller can for example be used to define the routes to be used for a given traffic flow based on observed packet delays and packet loss. However, if such decisions are solely based on the observed metrics (e.g. packet delay and packet loss) the decisions often come too late. Therefore, it is desirable to predict some network metrics (such as the packet delay and the packet loss) using a predictive model and based on the prediction take actions that would improve (e.g. reduce) the packet loss or the packet delay.

FIG. 6 illustrates a typical example of the prediction that can be made and the type of input that can be used for the prediction according to the present principles. In the example, the neural network model takes as input a network topology, a traffic matrix and network configuration and outputs a per-source-destination mean delay for the network.

In a variant of the present principles consists, the graph neural network predictive model already described is implemented in a SDN network controller. In this variant, the SDN controller has knowledge of the topology of the network it controls, i.e. the SDN controller stores or has access to information regarding the set of networking routers and the possible networking links between these routers. The SDN controller also knows the configured routes of the routers. Such information can be provided via manual configuration by an administrator, but it is also possible to use automatic means to provide the information to the SDN controller, such as letting the routers send messages to the SDN controller in order to inform the SDN controller of their presence. The routers may also announce the neighbouring routers to the SDN controller, thereby providing link information to the SDN controller. The routers may also announce their configured routes to the SDN controller, thereby providing a complete routing matrix to the SDN controller.

Each router may have parameters regarding its configuration which may be reported to the SDN central controller by the routers, such as:

- levelsQoS: number of supported QoS classes,
- queueSizes: queue sizes (in number of packets) for each output queue of the node, where values are separated by commas and where each value is associated with a QoS queue,
- schedulingPolicy: policy used to serve the QoS queues, the policy may be one of: 'FIFO' (First-In, First-Out), 'SP' (Strict Priority), 'WFQ' (Weighted Fair Queuing), and 'DRR' (Deficit Round Robin); if not specified, a 'FIFO' policy is considered by default, and
- schedulingWeights: list of weights separated by commas associated with each QoS queue; used only for the WFQ and DRR scheduling policies.

Each link may have parameters regarding its configuration or capacity to be collected by the SDN controller, such as:

- port: port number of the source node that links with destination node, and
- bandwidth: link bandwidth (bits/time unit).

Based on the information on routers (equivalent to “nodes”), links between routers (equivalent to “links”) and the routing matrix (equivalent to “path”) the SDN controller can construct the three graphs:

- $G_{p,l}$ connects each path to the links it uses,
- 5 • $G_{l,n}$ connects each link to the two nodes that define the link, and
- $G_{p,n}$ connects each path to the nodes it traverses.

The SDN controller can further collect measurements from the network infrastructure regarding traffic and network load. Routers may inform the SDN controller of existing network traffic and flows. Mechanisms and protocols such as netflow may for instance be used to this end. A traffic flow is characterized by source and destination (and a set of routers it traverses). Further, a flow may be characterized by bandwidth used, packet size distribution, etc.

The SDN controller can for instance collect at least some of the following flow-related information for each flow between a source node and a destination node:

- 15 • TimeDis: inter-packet arrival time distribution.
- TimeDistParams: dictionary with parameters of the specific inter-packet arrival time distribution:
 - EqLambda: average bitrate per time unit (bits/time unit).
 - AvgPktsLambda: average number of packets of average size generated per time unit (packets/time unit).
 - 20 ○ MinPktLambda: minimum number of packets generated per time unit (packets/time unit).
 - MaxPktLambda: maximum number of packets generated per time unit (packets/time unit).
 - 25 ○ StdDev: Standard deviation of the interarrival time (used in the normal distribution).
 - PktsLambdaOn: Average packets per time unit generated during the ON period (packets/time unit).
 - AvgTOff: Average duration of the OFF period. The duration of every OFF period is modeled with an exponential distribution of average =
 - 30 AvgTOff.

- AvgTOn: Average duration of the ON period. The duration of every ON period is modeled with an exponential distribution of average = AvgTOn.
 - ExpMaxFactor: Factor used to define an upper bound for exponential distributions. The upper bound is defined as: ExpMaxFactor * lambda of the exponential.
 - BurstGenLambda: Burst generation rate of the flow (bursts/time unit).
 - Bitrate: Bitrate during a burst: (bits/time unit).
 - ParetoMinSize: Minimum number of bits in the burst (bits).
 - ParetoMaxSize: Maximum number of bits in the burst (bits).
 - ParetoAlfa: Shape parameter (α) used in the Pareto distribution.
- SizeDist: packet size distribution used for the flow.
- SizeDistParams: dictionary with the parameters of the specific size distribution:
 - AvgPktSize: Average packet size (bits).
 - MinSize: Minimum packet size (bits).
 - MaxSize: Maximum packet size (bits).
 - PktSize1: First packet size option (bits).
 - PktSize2: Second packet size option (bits).
 - NumCandidates: Number of different packets size considered.
 - Size_i: Size of the candidate packet i (bits).
 - Prob_i: Probability to select candidate packet i (bits).
- AvgBw: average bandwidth for this flow (bits/time unit).
- PktsGen: packets generated by this flow per time unit (packets/time unit).
- ToS: Type of Service associated to this flow defined as an integer.

The SDN controller can also collect aggregated information regarding traffic measurements for a specific source and destination pair, for example:

- AvgBw: average bandwidth from node src to node dst (bits/time unit).
- PktsGen: packets generated from node src to node dst per time unit (packets/time unit).

The SDN controller can predict network metrics by using the described machine learning model, trained either on the same network and topology or on another network and topology.

For the network metric prediction, the SDN controller initializes each vertex of the graphs $G_{p,l}$, $G_{l,n}$, $G_{p,n}$ with the metrics and parameters collected by the SDN controller for each of the corresponding router, link or path. As has been described, the model employs message passing convolutions (i.e. passing the result of a vertex to its neighbors, each aggregating the received messages). The final prediction combines the resulting outputs using a fully connected layer.

The SDN controller may predict different kind of metrics, for example:

- PktsDrop: packets dropped per time unit over a path defined by a source and destination pair.
- AvgDelay: average per-packet delay over the packets transmitted on path defined by a source and destination pair.
- p10, p20, p50, p80, and p90: percentiles (10, 20, 50, 80, and 90) of the per-packet delay distribution on path defined by a source and destination pair.
- Jitter: Variance of the per-packet delay over the packets transmitted on path defined by a source and destination pair.

What-if scenarios

In a variant of the present principles, the SDN controller tests different paths using the prediction model. This can for example be done to simulate node failure, and different node or link characteristics.

The SDN controller may for instance test different routing matrices, constructing multiple versions of the graphs $G_{l,n}$ and $G_{p,n}$ and provide a network metric prediction for each routing matrix. By comparing the different prediction results, the SDN controller can pick the best result (e.g. the lowest latency) and reconfigure the communication network accordingly.

The SDN controller may for instance test different router configurations, e.g. by changing one or more of the parameters in the prediction model, such as levelsQoS, queueSizes, schedulingPolicy and schedulingWeights. This can be

achieved by initiating the hidden states with the tested values. Doing so, the SDN controller can generate predictions for each configuration and choose the configuration that provided the best prediction (e.g. lowest latency, lowest packet drop, smallest jitter). Based on this prediction, the SDN controller can reconfigure the
5 routers accordingly.

FIG. 7 illustrates a network analysis system according to an embodiment of the present principles. In the embodiment, the system 700 is centralized and includes a computing device 710 and a display 720 for displaying results. The computing device 710 includes a user interface 711 (that can be implemented with the display 720 as a
10 touchscreen) configured to receive user input, at least one hardware processor 712 configured to perform calculations described herein, memory 713 configured to store program code, (intermediate) results, etc. and an input/output (I/O) interface 714 configured for communication with external devices, for example network nodes. Also illustrated is a non-transitory computer-readable storage medium storing instructions
15 that, when executed, cause the at least one hardware processor to perform at least one method of the present principles.

Application to 3GPP communication networks

In a variant of the present principles, the graph neural network predictive model already described is implemented in the Network Data Analytics Function (NWDAF).
20 In this variant, the NWDAF has knowledge of the topology of the 3GPP network (e.g. the set of User Plane Functions (UPFs) of a 3GPP core network), i.e. the NWDAF controller stores or has access to information regarding the set of UPFs and the possible networking links between these UPFs. The NWDAF also knows the configured routes of the UPFs (e.g. the set of configured and used Protocol Data Unit
25 (PDU) sessions). Such information (e.g. the configured routes) may be provided by the Session Management Function (SMF), as it is the SMF defines and configures the PDU-session therefore defining routing policy.

It is also possible to use other automatic means to provide the information to the NWDAF, such as letting the UPF send messages to the NWDAF in order to inform
30 the NWDAF of its presence. The UPFs may also announce the neighbouring UPFs to the NWDAF, thereby providing link information to the NWDAF. The UPFs may also

announce their configured routes to the NWDAF, thereby providing a complete routing matrix to the NWDAF. Finally, such information may alternatively also be provided via manual configuration by an administrator.

Each UPF may have parameters regarding its configuration and load which may be reported to the NWDAF by the UPF, such as:

- number of flows handled,
- QoS Enforcement Rules (QER) applied by the UPF,
- UPF specific configuration rules (which may be vendor specific),
- queueSizes: queue sizes (in number of packets) for each output queue of the node, where values are separated by commas and where each value is associated with a QoS queue,
- schedulingPolicy: policy used to serve the QoS queues, the policy may be one of: 'FIFO' (First-In, First-Out), 'SP' (Strict Priority), 'WFQ' (Weighted Fair Queuing), and 'DRR' (Deficit Round Robin); if not specified, a 'FIFO' policy is considered by default,
- schedulingWeights: list of weights separated by commas associated with each QoS queue; used only for the WFQ and DRR scheduling policies, and
- traffic usage report (report of user plane traffic in the UPF for the accumulated usage of network resources (see TS 29.244 [17]))

Each link may have parameters regarding its configuration or capacity to be collected by the NWDAF, such as:

- port: port number of the source node that links with the destination node, and
- bandwidth: link bandwidth (bits/time unit).

Based on the information on UPFs (equivalent to "nodes"), links between UPFs (equivalent to "links") and the routing matrix (equivalent to "path") the NWDAF can construct the three graphs:

- $G_{p,l}$ connects each path to the links it uses,
- $G_{l,n}$ connects each link to the two nodes that define the link, and
- $G_{p,n}$ connects each path to the nodes it traverses.

The NWDAF can further collect measurements from the network infrastructure regarding traffic and network load. UPFs may inform the NWDAF of existing network

traffic and flows. The NWDAF may expose a specific API for this purpose. A traffic flow is characterized by source and destination (and a set of routers it traverses). Further, a flow may be characterized by bandwidth used, packet size distribution, etc.

The NWDAF can for instance collect at least some of the following flow-related information for each flow between a source node and a destination node:

- QoS parameters per flow according to 5G QoS Identifier (5QI):
 - Resource Type (Guaranteed Bit Rate (GBR), Delay critical GBR or Non-GBR);
 - Priority level;
 - Packet Delay Budget;
 - Packet Error Rate;
 - Averaging window (for GBR and Delay-critical GBR resource type only);
 - Maximum Data Burst Volume (for Delay-critical GBR resource type only).
- Flow bitrates:
 - Guaranteed Flow Bit Rate (GFBR) – Uplink (UL) and Downlink (DL);
 - Maximum Flow Bit Rate (MFBR) -- UL and DL.
- The NWDAF may also collect at least some of the following flow related information:

Information	Source	Description
QFI	SMF	QoS Flow Identifier.
QoS flow Bit Rate	UPF	The observed bit rate for UL direction; and The observed bit rate for DL direction.
QoS flow Packet Delay	UPF	The observed Packet delay for UL direction; and The observed Packet delay for the DL direction.
Packet transmission	UPF	The observed number of packet transmission.
Packet retransmission	UPF	The observed number of packet retransmission.
Measurement period	UPF or AF	Measurement period.
Throughput UL/DL	UPF or AF	Average Throughput UL/DL over the measurement period.
Throughput UL/DL (peak)	UPF or AF	Peak Throughput UL/DL over the measurement period.
Timestamp	UPF or AF	Time when measurements are taken.
Achieved sampling ratio	UPF	Sampling ratio achieved by UPF.

The NWDAF can predict network metrics by using the described machine learning model, trained either on the same network and topology or on another network and topology.

For the network metric prediction, the NWDAF initializes each vertex of the graphs $G_{p,l}$, $G_{l,n}$, $G_{p,n}$ with the metrics and parameters collected by the NWDAF for each of the corresponding router, link or path. As has been described, the model employs message passing convolutions (i.e. passing the result of a vertex to its neighbors, each aggregating the received messages). The final prediction combines the resulting outputs using a fully connected layer.

The NWDAF may predict different kind of metrics, for example:

- PktsDrop: packets dropped per time unit over a path defined by a source and destination pair.
- AvgDelay: average per-packet delay over the packets transmitted on a path defined by a source and destination pair.

- p10, p20, p50, p80, and p90: percentiles (10, 20, 50, 80, and 90) of the per-packet delay distribution on a path defined by a source and destination pair.
- Jitter: Variance of the per-packet delay over the packets transmitted on a path defined by a source and destination pair.

FIG. 8 illustrates a method in a 5G network according to the present principles.

In step S810, a SMF 801 provides network configuration information to a NWDAF 803. The network configuration information can relate to the currently deployed network or to a non-existent network to be tested. As explained, the network configuration information can for example include links, routes, PDU session, load information and router configurations.

In alternate or complementary step S820, a UPF 805 provides network information to the NWDAF 803. As explained, the network information can for example include information about the UPF 805, configurations, links, neighboring UPF and network load information.

In step S830, the NWDAF uses the received information to generate a model, for example a GNN model, as explained.

In step S840, the SMF 801 sends a request for a performance prediction to the NWDAF 803. The request can for example relate to delays in the network.

In step S850, the NWDAF 803 predicts, based on information received in for example step S810 and/or step S820, a performance of the network to answer the request.

In step S860, the NWDAF 803 sends a corresponding prediction result to the SMF 801.

In step S870, acting on the prediction result, the SMF 801 can, change or request changes of the UPF configuration based on the prediction result.

What-if scenarios

In a variant of the present principles, a networking function (e.g. the SMF) tests different paths using the prediction service exposed by the NWDAF. This can for example be done to simulate different choices before assigning a PDU-session to one
 5 or another UPF.

The SMF submits a prediction request to the NWDAF with at least one configuration (e.g. terminating a PDU-session on UPF A). The NWDAF response with a prediction of a network performance metric. Based on this response the SMF decides to apply this configuration, to test another configuration or to do nothing.

10 **Distributed implementation**

So far, the present principles have been described for a centralized implementation, that is, an implementation where a single computing architecture (e.g. computing device such as a computer) performs the computations. In practice, the invention may be fully or partially implemented directly in the physical
 15 network/topology, in a distributed setting. Possible advantages of this include a reduced the number of messages transmitted by the network (e.g. to provide network metrics or configurations) and removal of the need for a potentially large bandwidth at the central server. A distributed implementation may also be desirable if the invention is applied to an infrastructure owned by multiple operators, as this would
 20 limit information shared between competing operators. It will be appreciated that in a distributed implementation, each computing architecture can be implemented essentially like the system illustrated in FIG. 7.

To illustrate this, two possible implementations will be described: a partially distributed one, and a completely distributed one.

25 Three embeddings are computed: link embeddings $\mathbf{l}$, node embeddings $\mathbf{n}$ and path embeddings $\mathbf{p}$.

Partially distributed implementation

In this implementation, only link and node embeddings are distributed over the computing resources c located in the nodes of the physical/target network. Different
 30 embeddings can be allocated to each computing resource. Then, each resource c_i is responsible for the update of the embeddings and for the transmission of the updated

embeddings to a central processing architecture α and to other relevant computing resources.

The distributed implementation can work as follows:

1. Initialization:

- 5 a. Each resource c_i is attributed the embedding n_i of the associated topological node.
- b. Each link embedding l_j is attributed to a resource c_i among the resources that have been associated to the source and/or the destination of the link.
- 10 c. Parameters of the graph neural network are sent to the resources (e.g. the trained model, hyperparameters etc).
- d. Each resource c_i stores, for each embeddings it is responsible for, the resources to which c_i should send an updated value of the embedding.
 - 15 i. For node embedding, these are all resources responsible for links originating or ending in that node (iterative process)
 - ii. For link embedding, these are all computing resources that are responsible for a node in that link and/or a path containing that link (iterative process)
 - 20 iii. For path embedding, these are all computing resources that are responsible for a link in that path (iterative process) and all nodes contained in that path (final computation)

2. Computation:

- a. The described inference algorithm. Every step in the algorithm that consists in the computation of an embedding is extended as follows.

25 After an embedding has been computed by a resource c_i , the value of this embedding is sent to the appropriate computing resources. Before computing an embedding, a resource c_i waits to receive the necessary values.

30 For step 1.a: if any topological node does not have an associated computing resource, this node is associated with another computing resource. This can for example be performed directly by the central processing architecture, based on node proximity etc.

In step 1.b: link embedding attribution can either be performed directly by the central processing architecture; by a specific rule, for example the resource responsible for the source (alternatively, the destination) of a link is responsible for the embedding of that link; by communication and agreement between the nodes; or
 5 by any other appropriate method.

Step 1.c: These parameters may for example be loaded from memory or may be broadcasted by the central computing resource a .

Step 1.d corresponds to the distribution of the structure of the three bipartite graphs.

10 Step 2.a can be done synchronously: the main algorithm moves to the next step only when all embeddings of a given type have been updated. It can also be done asynchronously: as soon as a resource c_i has received all (alternatively, some) values (of other embeddings) necessary to update one embedding it is responsible for, it computes this embedding and sends it to the appropriate computing resources.

15 Fully distributed implementation

In this implementation, the computation of the embeddings is distributed. The distributed implementation could work as follows:

1. Initialization:

- a. Each resource c_i is attributed the embedding n_i of the associated
 20 topological node.
- b. Each link embedding l_j is attributed to a resource c_i among the resources that have been associated to the source and/or the destination of the link.
- c. Each path embedding p_k is attributed to a resource c_i among the
 25 resources that have been associated to at least one node or link in the path.
- d. Parameters of the graph neural network are loaded by the resources.
- e. Each resource c_i stores, for each of the embeddings it is responsible for, the resources to which c_i must send an updated value of the
 30 embedding.

- i. For node embedding, these are all resources responsible for links originating or ending in that node (iterative process).
- ii. For link embedding, these are all computing resources that are responsible for a node in that link and/or a path containing that link (iterative process).
- iii. For path embedding, these are all computing resources that are responsible for a link in that path (iterative process) and all nodes contained in that path (final computation).

2. Computation:

- a. The described inference algorithm. Every step in the algorithm that consists in the computation of an embedding is extended as follows. After an embedding has been computed by a resource c_i , the value of this embedding is sent to the appropriate computing resources. Before computing an embedding, a resource c_i waits to receive the necessary values.

Step 1.c can for example be performed according to a specific rule, for example the resource responsible for the largest number of nodes or links in the path is responsible for the embedding of the path; resource responsible for similar paths are more likely to be responsible for a path; or by any other appropriate method.

It will thus be appreciated that the present principles can be used to provide solutions for network analysis that, for example, can predict performance or evaluate what-if scenarios.

It should be understood that the elements shown in the figures may be implemented in various forms of hardware, software or combinations thereof. Preferably, these elements are implemented in a combination of hardware and software on one or more appropriately programmed general-purpose devices, which may include a processor, memory and input/output interfaces.

The present description illustrates the principles of the present disclosure. It will thus be appreciated that those skilled in the art will be able to devise various arrangements that, although not explicitly described or shown herein, embody the principles of the disclosure and are included within its scope.

5 All examples and conditional language recited herein are intended for educational purposes to aid the reader in understanding the principles of the disclosure and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions.

10 Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosure, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function,
15 regardless of structure.

Thus, for example, it will be appreciated by those skilled in the art that the block diagrams presented herein represent conceptual views of illustrative circuitry embodying the principles of the disclosure. Similarly, it will be appreciated that any flow charts, flow diagrams, and the like represent various processes which may be
20 substantially represented in computer readable media and so executed by a computer or processor, whether or not such computer or processor is explicitly shown.

The functions of the various elements shown in the figures may be provided through the use of dedicated hardware as well as hardware capable of executing software in association with appropriate software. When provided by a processor, the
25 functions may be provided by a single dedicated processor, by a single shared processor, or by a plurality of individual processors, some of which may be shared. Moreover, explicit use of the term “processor” or “controller” should not be construed to refer exclusively to hardware capable of executing software, and may implicitly include, without limitation, digital signal processor (DSP) hardware, read only memory
30 (ROM) for storing software, random access memory (RAM), and non-volatile storage.

Other hardware, conventional and/or custom, may also be included. Similarly, any switches shown in the figures are conceptual only. Their function may be carried out through the operation of program logic, through dedicated logic, through the interaction of program control and dedicated logic, or even manually, the particular
5 technique being selectable by the implementer as more specifically understood from the context.

In the claims hereof, any element expressed as a means for performing a specified function is intended to encompass any way of performing that function including, for example, a) a combination of circuit elements that performs that function
10 or b) software in any form, including, therefore, firmware, microcode or the like, combined with appropriate circuitry for executing that software to perform the function. The disclosure as defined by such claims resides in the fact that the functionalities provided by the various recited means are combined and brought together in the manner which the claims call for. It is thus regarded that any means that can provide
15 those functionalities are equivalent to those shown herein.

CLAIMS

1. A method comprising:

obtaining a network model for predicting a characteristic of a network, the network model comprising three bipartite graphs corresponding to a topology of the network, wherein a first bipartite graph connects each path in the network to links the path uses in the network, a second bipartite graph connects each link in the network to nodes that define the link, and a third bipartite graph connects each path in the network to the nodes that define the path, wherein each vertex of the bipartite graphs contains a state;

injecting parameters of the network and load in the network into the network model;

performing T iterations of updating the states of at least some of the vertices based on current states of the vertices and states of neighbour vertices;

combining the state of every path to obtain a prediction of the characteristic;

and

outputting the prediction of the characteristic.

2. The method of claim 1, wherein performing T iterations of updating the states of at least some of the vertices based on current states of the vertices and states of neighbour vertices comprises:

updating the state of every link in the first bipartite graph as a combination of its current state and an aggregation of states of its neighbouring paths;

updating the state of every node in the second bipartite graph as a combination of its current state and an aggregation of states of its neighbouring links;

updating the state of every link in the second bipartite graph as a combination of its current state and an aggregation of states of its neighbouring nodes; and

updating the state of every path in the first bipartite graph as a combination of its current state and an aggregation of states of its neighbouring links; and

wherein combining the state of every path comprises, after the T iterations, updating the state of every path in the third bipartite graph as a combination of its current state and an aggregation of states of its neighbouring nodes.

3. The method of claim 1, wherein the network model is a graph neural network.

4. The method of claim 3, wherein the graph neural network is implemented in a data analytics function of a telecommunication network, wherein the graph neural network
5 represents a topology of the telecommunication network.

5. The method of claim 4, wherein the data analytics function is a NWDAF in a 3GPP network.

10 6. The method of claim 4, wherein a node in the graph neural network corresponds to a User Plane Function, UPF, in the telecommunication network.

7. The method of claim 6, wherein the data analytics function stores information about possible links between the UPFs.

15

8. The method of claim 6, wherein the data analytics function receives reports from UPFs, a report comprising at least one of, number of flows handled by the UPF, Quality of Service Enforcement Rules applied by the UPF, UPF configuration rules, queue sizes for each output queue of the UPF, a policy used to serve Quality of
20 Service queues of the UPF, user plane traffic in the UPF.

9. The method of claim 6, wherein the data analytics function receives, from a Session Management Function, SMF, at least one report concerning a UPF, the report comprising at least one of: a number of flows handled by the UPF, Quality of Service
25 Enforcement Rules applied by the UPF, UPF configuration rules, queue sizes for each output queue of the UPF, a policy used to serve Quality of Service queues of the UPF, user plane traffic in the UPF.

10. The method of claim 4, wherein a characteristic is one of packets dropped per
30 time unit over a path defined by a source and destination pair, average per-packet delay over the packets transmitted on path defined by a source and destination pair, percentiles of per-packet delay distribution on a path defined by a source and

destination pair, variance of per-packet delay over packets transmitted on a path defined by a source and destination pair.

11. The method of claim 1, wherein the network model corresponds to an existing
5 network.

12. An apparatus comprising memory and a processor configured to:

obtain a network model for predicting a characteristic of a network, the network
model comprising three bipartite graphs corresponding to a topology of the network,
10 wherein a first bipartite graph connects each path in the network to links the path uses
in the network, a second bipartite graph connects each link in the network to nodes
that define the link, and a third bipartite graph connects each path in the network to
the nodes that define the path, wherein each vertex of the bipartite graphs contains a
state;

15 inject parameters of the network and load in the network into the network
model;

perform T iterations of updating the states of at least some of the vertices
based on current states of the vertices and states of neighbour vertices;

combine the state of every path to obtain a prediction of the characteristic; and
20 output the prediction of the characteristic.

13. The apparatus of claim 12, wherein being configured to perform T iterations of
updating the states of at least some of the vertices based on current states of the
vertices and states of neighbour vertices comprises being configured to:

25 update the state of every link in the first bipartite graph as a combination of its
current state and an aggregation of states of its neighbouring paths;

update the state of every node in the second bipartite graph as a combination
of its current state and an aggregation of states of its neighbouring links;

update the state of every link in the second bipartite graph as a combination of
30 its current state and an aggregation of states of its neighbouring nodes; and

update the state of every path in the first bipartite graph as a combination of its
current state and an aggregation of states of its neighbouring links; and

wherein being configured to combine the state of every path comprises, after the T iterations, comprises being configured to update the state of every path in the third bipartite graph as a combination of its current state and an aggregation of states of its neighbouring nodes.

5

14. The apparatus of claim 12, wherein the network model is a graph neural network.

15. The apparatus of claim 14, wherein the apparatus is a data analytics function of a telecommunication network, wherein the graph neural network represents a topology of the telecommunication network.

10

16. The apparatus of claim 15, wherein the data analytics function is a NWDAF in a 3GPP network.

17. The apparatus of claim 15, wherein a node in the graph neural network corresponds to a User Plane Function, UPF, in the telecommunication network.

15

18. The apparatus of claim 17, wherein the data analytics function stores information about possible links between the UPFs.

20

19. The apparatus of claim 17, wherein the data analytics function receives reports from UPFs, a report comprising at least one of, number of flows handled by the UPF, Quality of Service Enforcement Rules applied by the UPF, UPF configuration rules, queue sizes for each output queue of the UPF, a policy used to serve Quality of Service queues of the UPF, user plane traffic in the UPF.

25

20. The apparatus of claim 17, wherein the data analytics function receives, from a Session Management Function, SMF, at least one report concerning a UPF, the report comprising at least one of: a number of flows handled by the UPF, Quality of Service Enforcement Rules applied by the UPF, UPF configuration rules, queue sizes for each output queue of the UPF, a policy used to serve Quality of Service queues of the UPF, user plane traffic in the UPF.

30

21. The apparatus of claim 15, wherein a characteristic is one of packets dropped per time unit over a path defined by a source and destination pair, average per-packet delay over the packets transmitted on path defined by a source and destination pair, percentiles of per-packet delay distribution on a path defined by a source and destination pair, variance of per-packet delay over packets transmitted on a path defined by a source and destination pair.

22. The apparatus of claim 12, wherein the network model corresponds to an existing network.

10

23. A non-transitory computer-readable storage medium storing instructions that, when executed, cause at least one hardware processor to perform a method of claim 1.

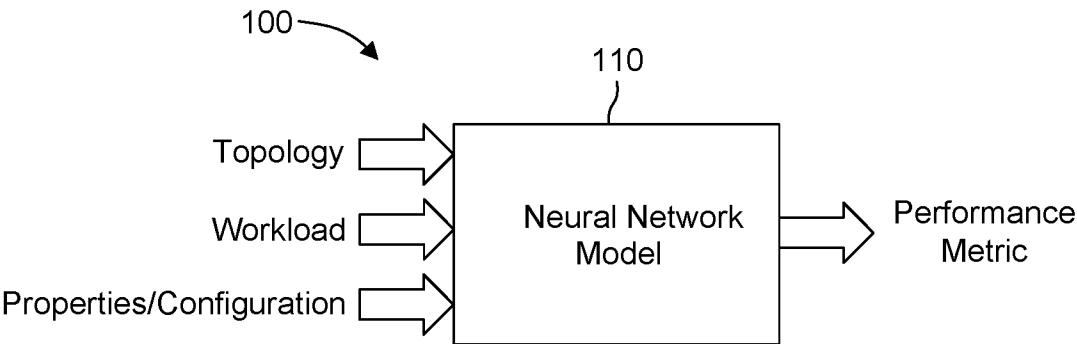


FIG. 1

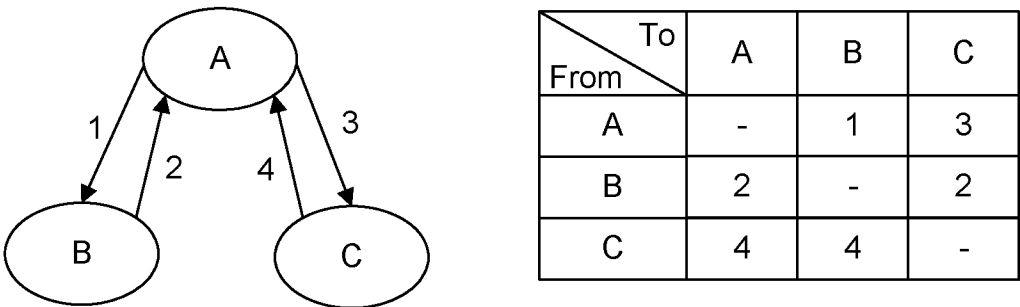


FIG. 2

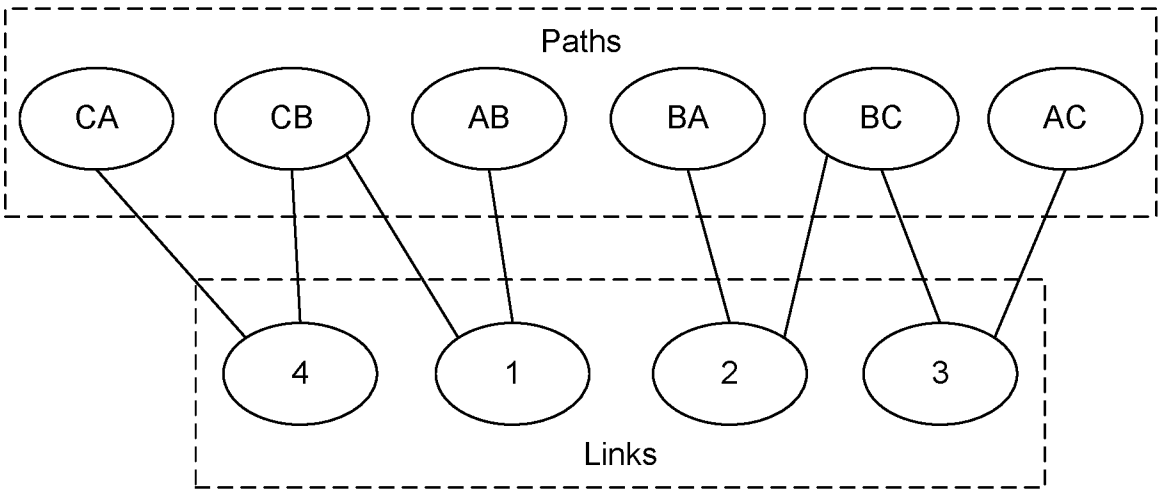


FIG. 3

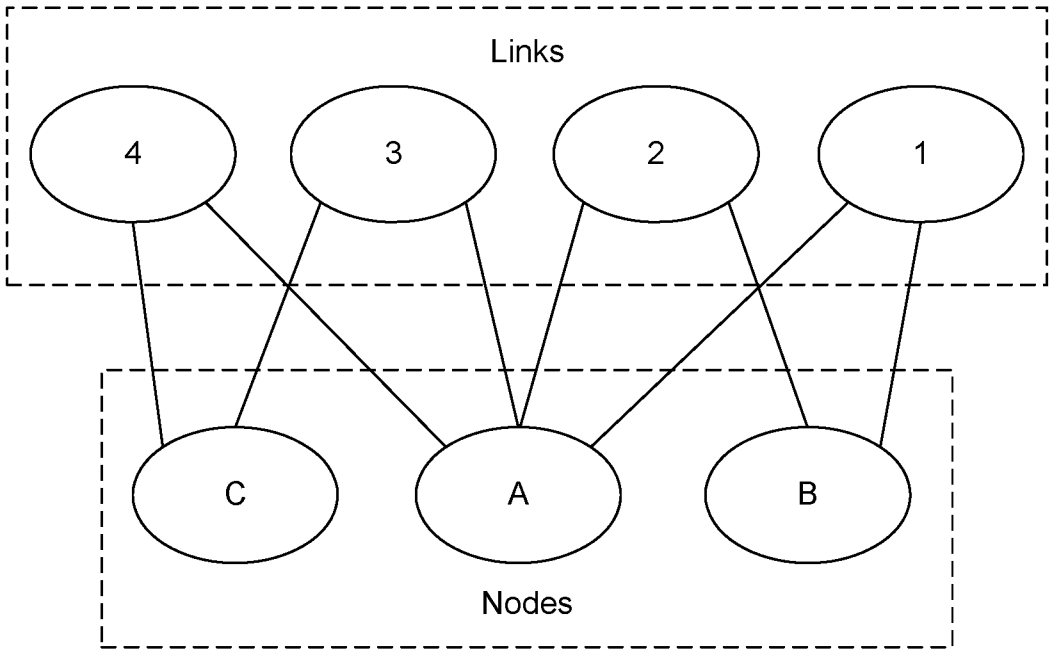


FIG. 4

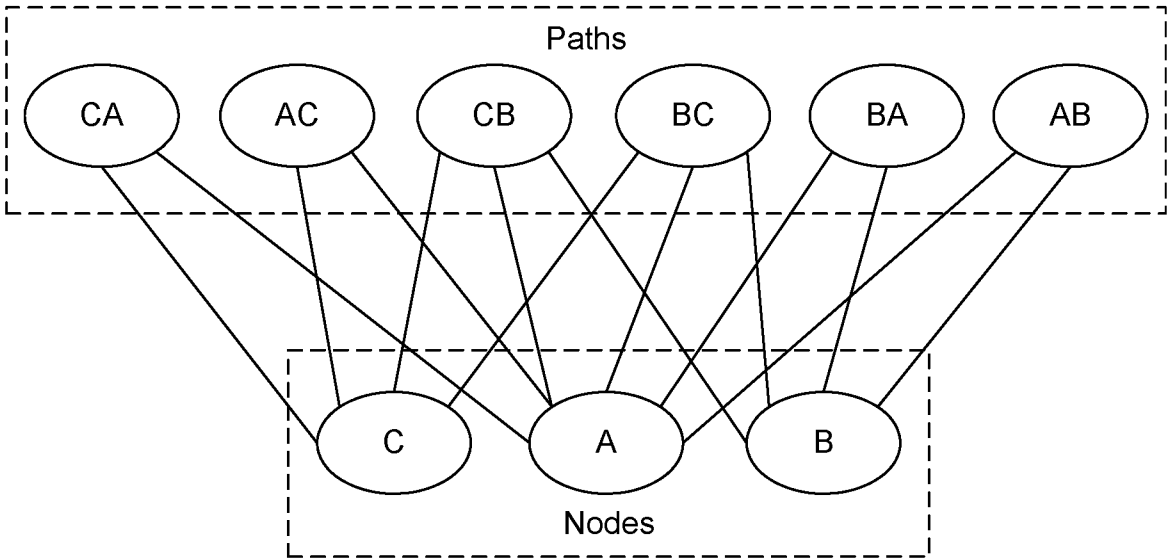


FIG. 5

3/4

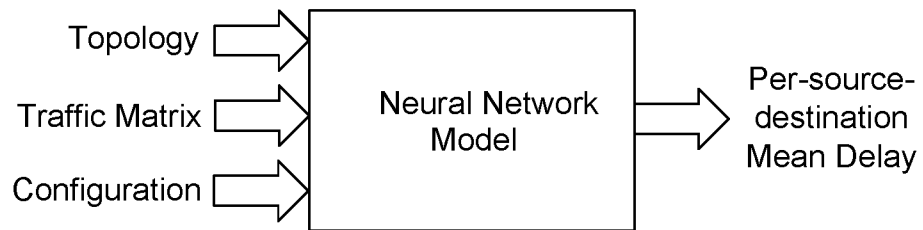


FIG. 6

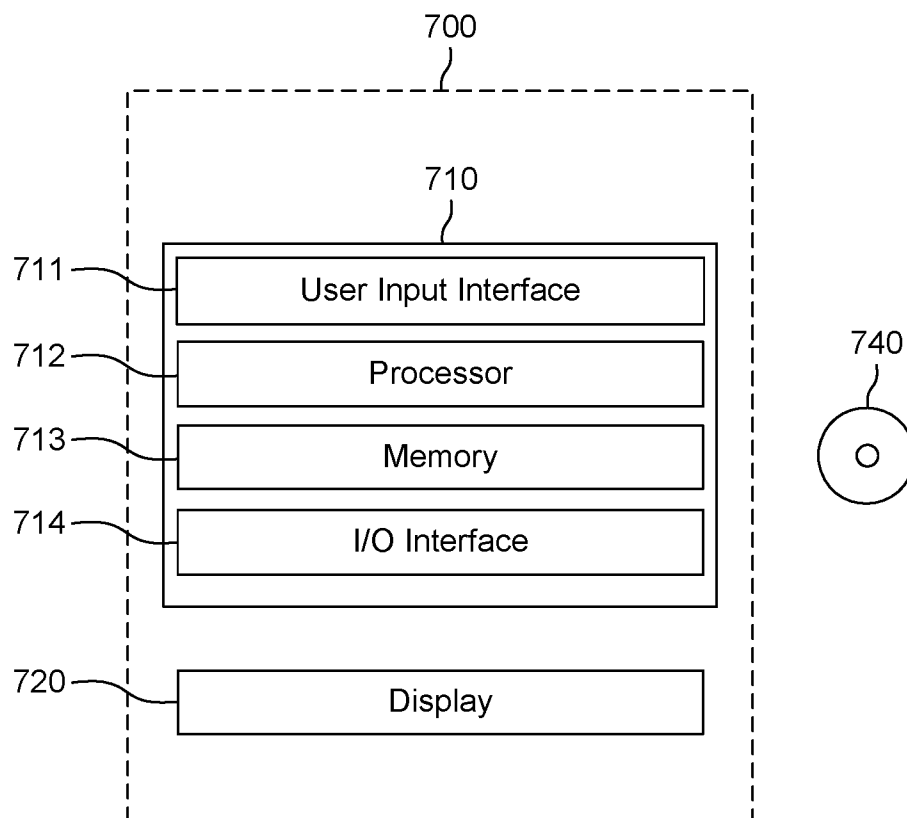


FIG. 7

4/4

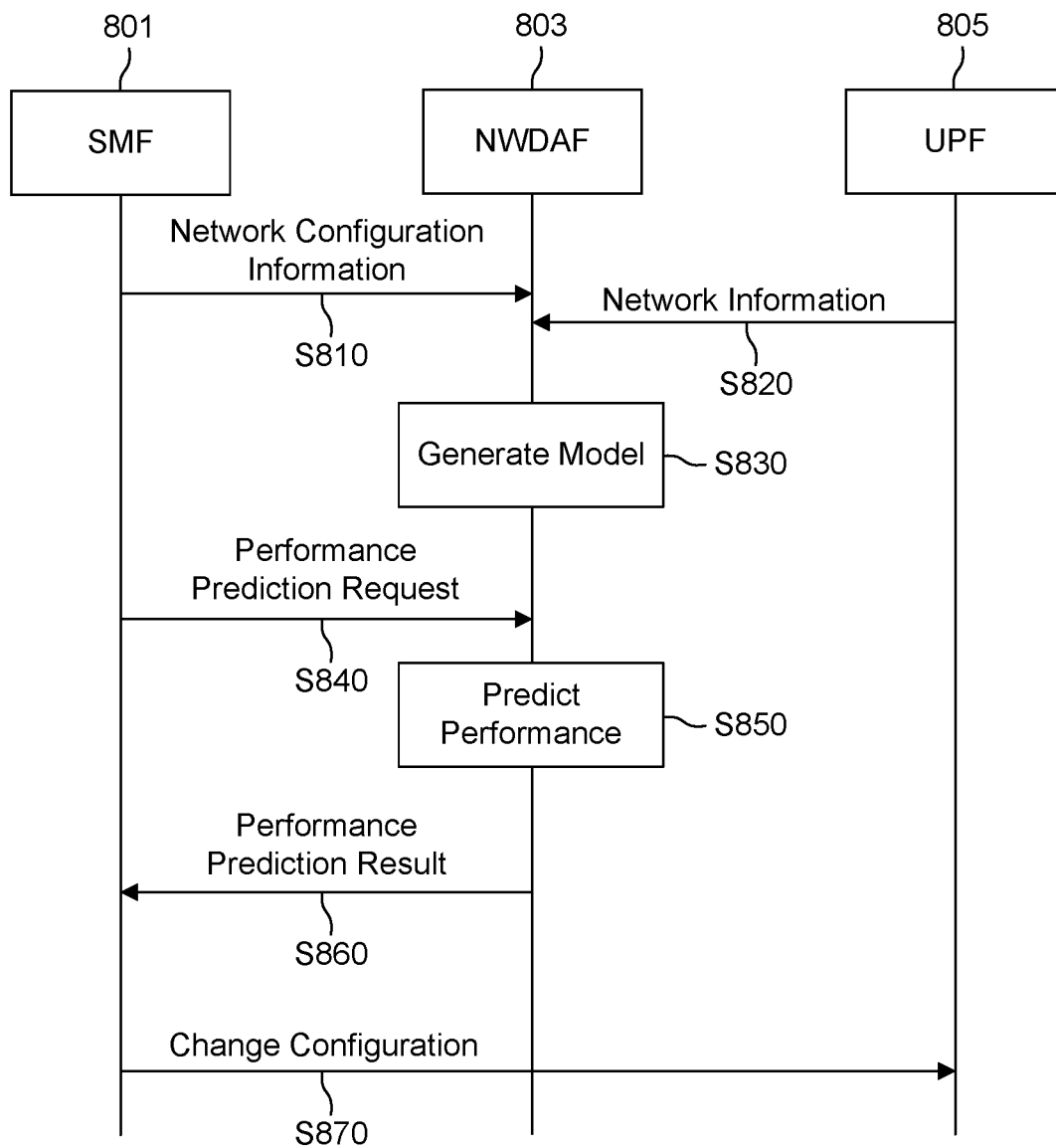


FIG. 8

International application No
PCT/EP2021/081460

INV.	H04L41/147	H04L41/16		
ADD.	H04L41/0823	H04L43/0852	H04L41/122	H04L41/149

B. FIELDS SEARCHED

H04L

EPO-Internal, WPI Data

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KRZYSZTOF RUSEK ET AL: "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 28 October 2019 (2019-10-28), XP081932165, DOI: 10.1145/3314148.3314357	1, 3-12, 14-23
A	abstract sections 1.2, 2, 3.1, 3.2, 4.2 figure 2 ----- -/--	2, 13

x See patent family annex.

"&" document member of the same patent family

03/03/2022

Ramenzoni, Stefano

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2021/081460

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	MING GAO ET AL: "Learning Vertex Representations for Bipartite Networks", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 16 January 2019 (2019-01-16), XP081626090, sections 1 to 3, 3.1 -----	1-23
A	CN 111 314 171 A (SHENZHEN POWER SUPPLY BUREAU) 19 June 2020 (2020-06-19) paragraph [0003] - paragraph [0013] paragraph [0053] - paragraph [0062] -----	1-23
A	US 2014/317033 A1 (MOJSILOVIC ALEKSANDRA [US] ET AL) 23 October 2014 (2014-10-23) paragraphs [0001], [0008], [0023] - [0030], [0054] - [0059], [0064] -----	1-23
A	US 2019/349287 A1 (CHANDRA SEKAR RAO VENKATA [IN] ET AL) 14 November 2019 (2019-11-14) paragraphs [0004], [0005], [0028], [0035], [0037] - [0039], [0046] - [0048] -----	1, 12, 23

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2021/081460

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
CN 111314171	A	19-06-2020	NONE	
US 2014317033	A1	23-10-2014	US 2014317033 A1	23-10-2014
			US 2014317038 A1	23-10-2014
US 2019349287	A1	14-11-2019	NONE	