

Управление качеством сервиса в сетях

Применение методов машинного обучения для обеспечения требуемого качества сервиса

Степанов Евгений Павлович,
к.ф.-м.н., м.н.с.







Best effort

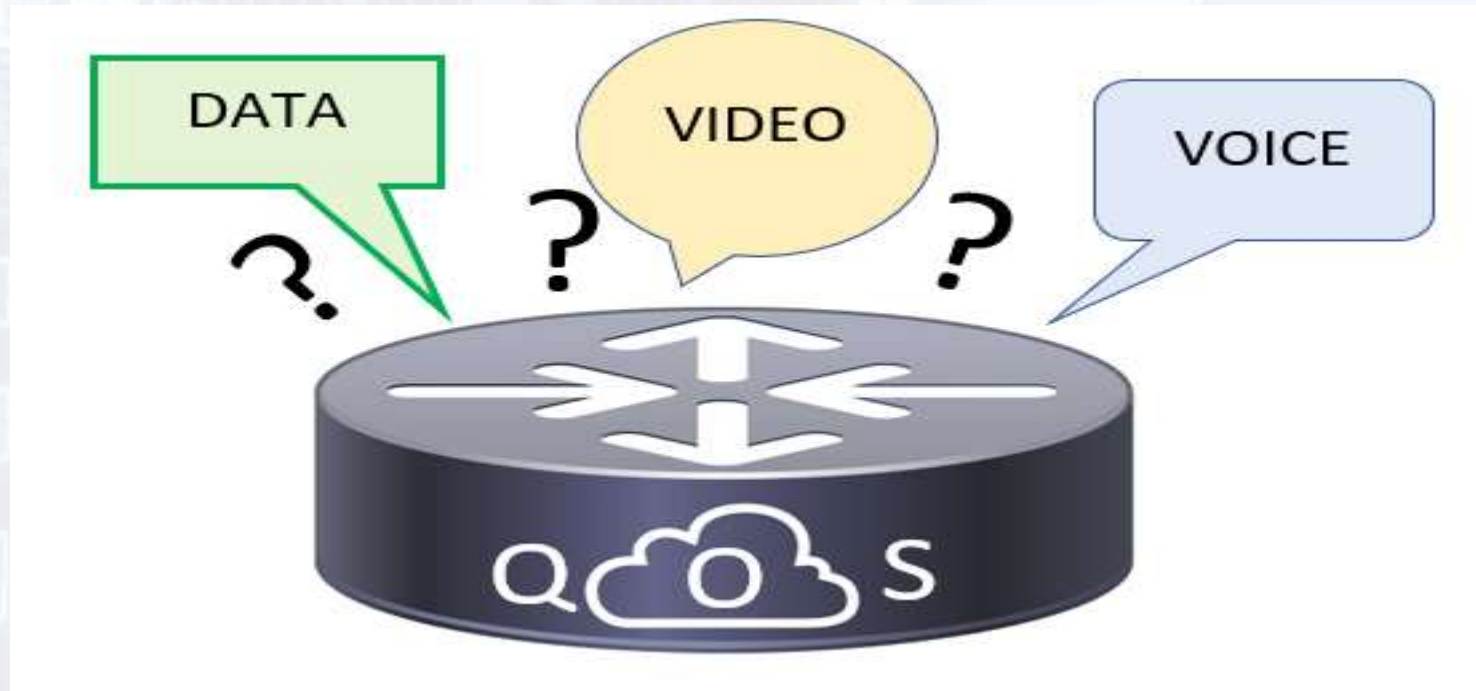
- скорость соединения с Интернет в самом лучшем случае В Мбит/с.
- если не нравится, переключайся на другого провайдера



QoS



- некоторые гарантии по пропускной способности, задержке...



Метрики качества



Пропускная способность

Количество данных, которые может быть передано в единицу времени [байты в секунду]



Метрики качества

Задержка

Время передачи единицы данных по направлению от отправителя к получателю [секунды]

Очень долгое
время обслуживания

Как долго это займет?

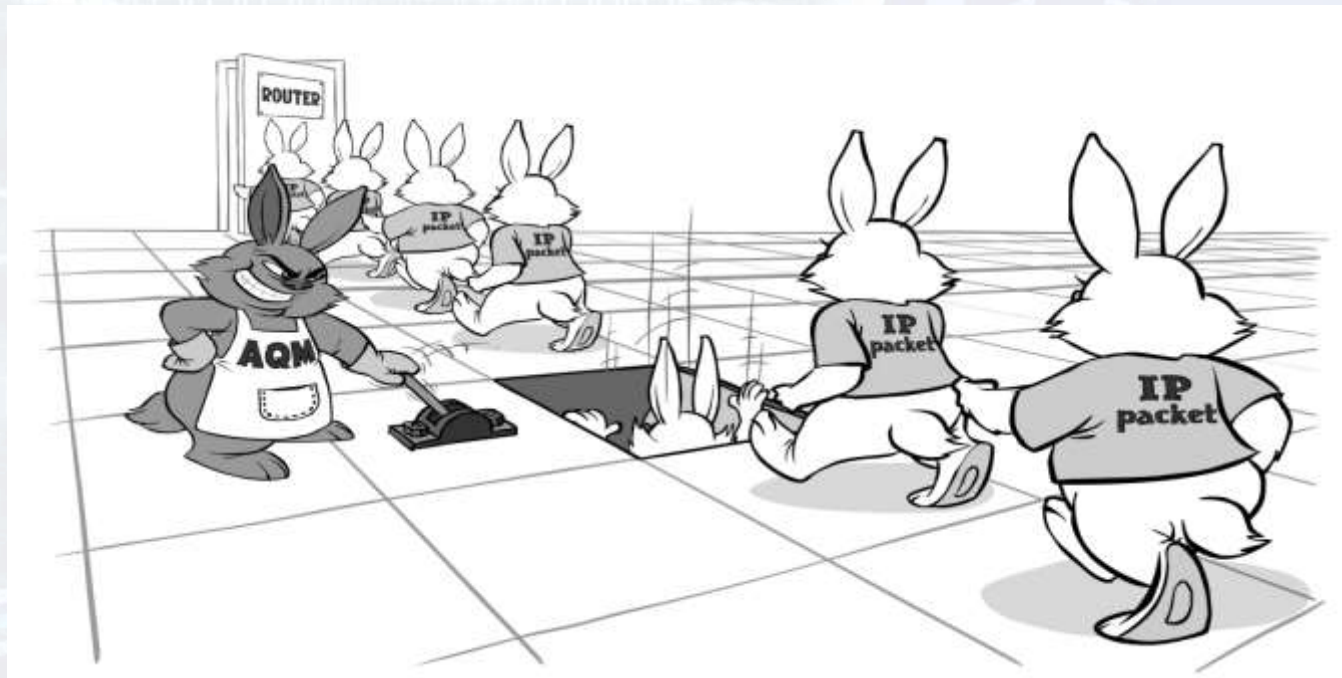
Ждать или нет?



Метрики качества

Уровень потерь

Доля пакетов, которые были отправлены, но не были доставлены получателю
[проценты]



Почему QoS становится всё более важной проблемой?



Почему QoS становится всё более важной проблемой?



Почему QoS становится всё более важной проблемой?



Почему QoS становится всё более важной проблемой?



Почему QoS становится всё более важной проблемой?



How to shorten OTT TV Lag - Ops Perspective

IPTV OTT Lag 4/5 – QoS and Ops Analytics

- 33% Abandonment rate immediately with poor quality *Akamai
- 50% Abandonment rate with 5 startup sec delay *Akamai
- 66% Consumers are frustrated with video buffering *Research Now
- 21% Severely irritated when a video didn't load quickly *Research Now
- 38% Users heart rate increase when buffering *Ericsson



David Murargi, Portugal Telecom - 09.11.2016

**Very High Demanding Customers,
Public Market Rating and Posts,
Low Barriers to Churn**

Path

Customer Care:

OTT Users won't call u as before, they do public complains in Public Stores, blogs, forums and social media. **Be proactive in capturing the Voice of Customer** and react as fast as possible in your devops cycle.

Tech Level :

Measure, Measure, Measure Experience, Engagement and Audience in Real Time to automatically prevent problems and monetize!

We've done the path in IPTV. In OTT it's harder as you don't own the entire chain.

Local Operators can do it better because they own the network and have direct access to the customers

Управление качеством сервиса



Как управлять качеством сервиса?



Подходы:

- 1. Управление перегрузкой**
 - Современные протоколы управления перегрузкой TCP
- 2. Демультимплексирование/мультиплексирование**
 - Многопоточные транспортные протоколы
 - Маршрутизация на уровне интернет провайдеров
 - Network Coding
- 3. Сегментация**
 - TCP Proxy
- 4. Балансировка**
 - Балансировка нагрузки и управление трафиком
- 5. Преобразование сообщений**
 - FEC
 - Сжатие

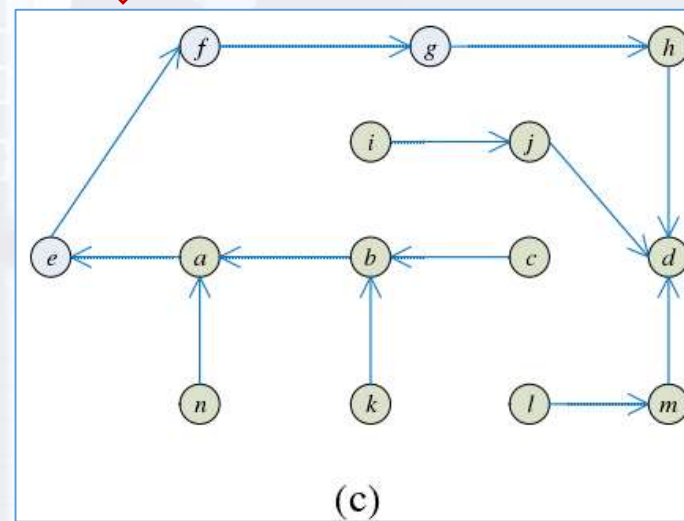
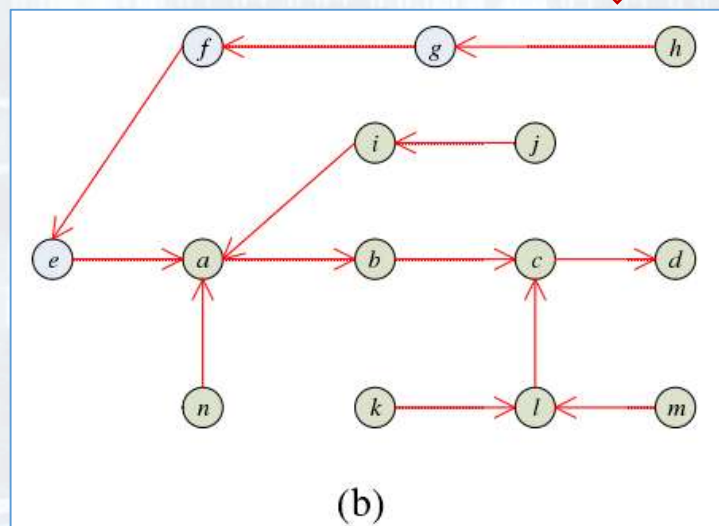
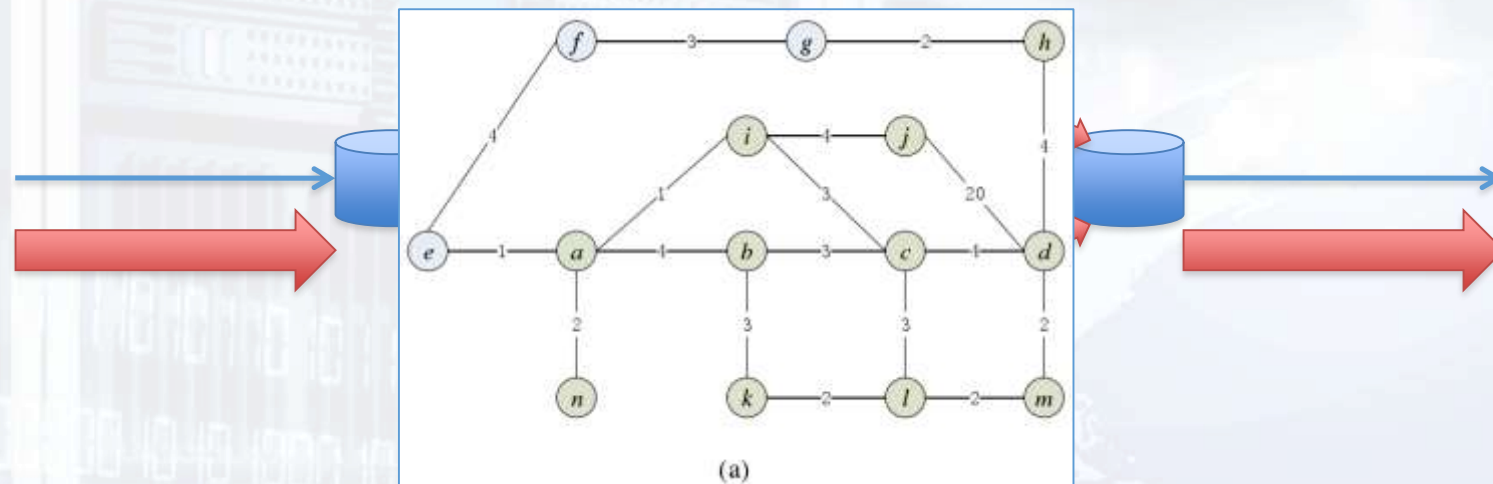
Модели оценки качества сервиса:

- Активные и пассивные измерения
- Сетевое исчисление: математический подход к качеству сервиса
- NS3: моделирование поведения сети с высокой точностью

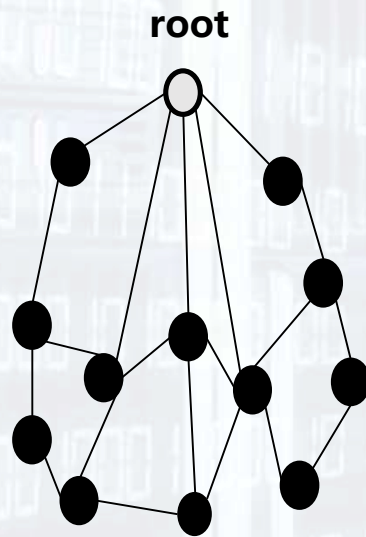
Примеры:

- Управление сетевыми ресурсами в Центрах Обработки Данных
- Обеспечение качества сервиса в сетях доставки контента
- Пропускная способность по требованию

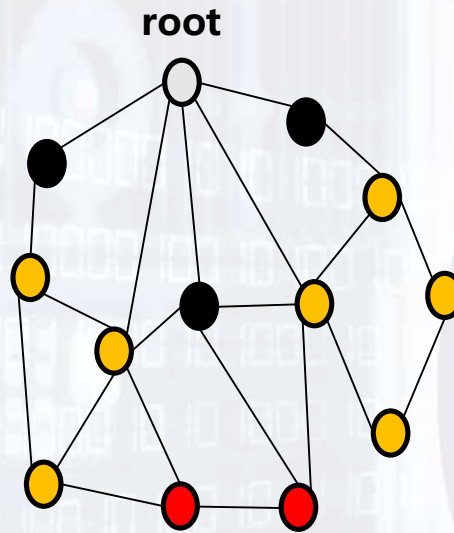
Балансировка нагрузки



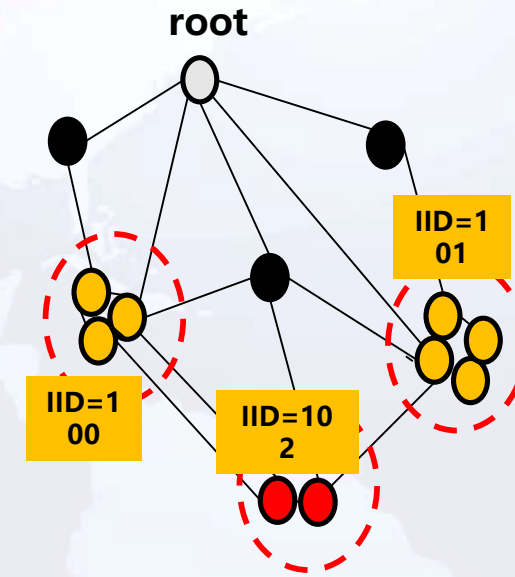
Балансировка нагрузки / Indirect Next Hop



Original graph



Partition graph

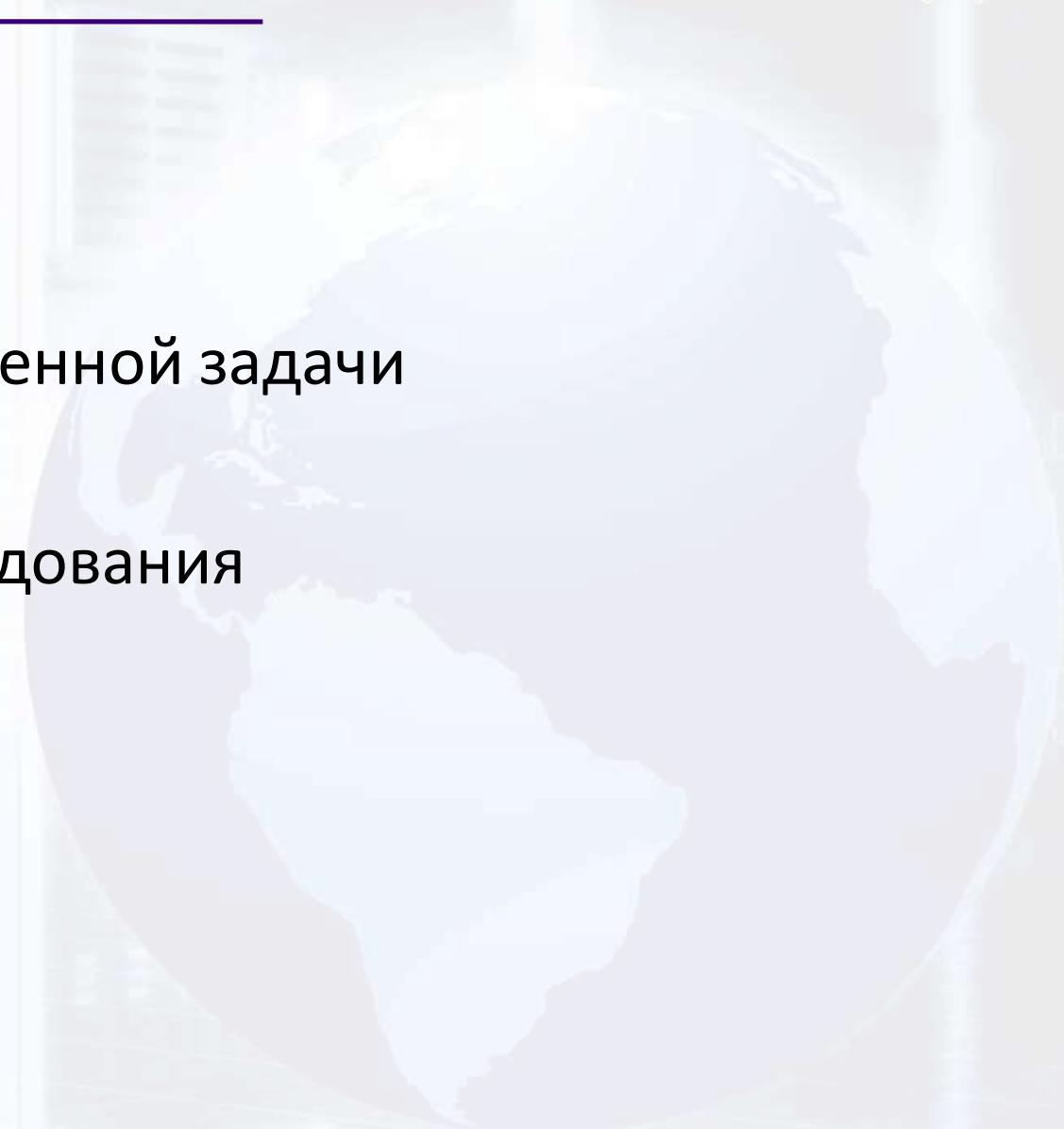


Node in one group share same IID

Этапы исследования



- Математическая постановка задачи
- Проведение обзора схожих работ
- Разработка алгоритма решения поставленной задачи
- Реализация разработанного алгоритма
- Проведение экспериментального исследования



Постановка задачи



Какие входные данные алгоритма?

Network Topology: mathematical description

1 Basic Definitions

Recall some definitions of Graph Theory [1]. An undirected graph G is an ordered pair (V, E) consisting of a nonempty finite set V of vertices and a finite set E , disjoint from V , of edges, together with an incidence function ι_E that associates with each edge $e \in E$ an unordered pair $\{v, w\}$ of (not necessarily distinct) vertices $v, w \in V$. For simplicity, if $\iota_E(e) = \{v, w\}$ then we write $e = (v, w)$. We denote the set of vertices and the set of edges in G by $V(G)$ and $E(G)$, respectively.

If $e = (v, w) \in E$, where $v, w \in V$, then we say that e joins v and w , and the vertices v and w are called the ends of e . The ends of an edge are said to be incident with the edge, and vice versa. Two vertices which are incident with a common edge are adjacent, as are two edges which are incident with a common vertex, and two distinct adjacent vertices are neighbors. An edge with identical ends is called a loop, and an edge with distinct ends a link. Two or more links with the same pair of ends are said to be parallel edges.

A graph F is called a subgraph of a graph G if $V(F) \subseteq V(G)$, $E(F) \subseteq E(G)$, and ι_F is the restriction of ι_G to $E(F)$. We then say that F is a subgraph of itself, if induced subgraphs of G : if V' is a subset of V whose edge set consists of all edges with one end in X and the other end in Y , where $X, Y \subseteq V$ and $X \cap Y = \emptyset$, then we say that F is a bipartite graph with parts X and Y and with the set of edges E_X , where

By a **fat tree** we mean a single $k/2$ -partite graph $G = (V, E)$ with parts X, Y, Z, E such that:

- $|X| = k/2, |Y| = k/2, |Z| = k/2, |E| = k/2$.
- the set X can be partitioned into subsets X_1, \dots, X_k , where the vertices of X_s can be marked as $x_{s,1}, \dots, x_{s,2}, \dots, x_{s,k/2}$, i.e. $X_s = \{x_{s,1}, \dots, x_{s,k/2}\}$, $s = 1, \dots, k/2$.
- the set Y can be partitioned into subsets Y_1, \dots, Y_k , where the vertices of Y_s can be marked as $y_{s,1}, \dots, y_{s,2}, \dots, y_{s,k/2}$, i.e. $Y_s = \{y_{s,1}, \dots, y_{s,k/2}\}$, $s = 1, \dots, k/2$.
- the set Z can be partitioned into subsets Z_1, \dots, Z_k , where the vertices of Z_s can be marked as $z_{s,1}, \dots, z_{s,2}, \dots, z_{s,k/2}$, i.e. $Z_s = \{z_{s,1}, \dots, z_{s,k/2}\}$, $s = 1, \dots, k/2$.
- the set E can be partitioned into subsets E_1, \dots, E_k , where the vertices of E_s can be marked as $e_{s,1}, \dots, e_{s,2}, \dots, e_{s,k/2}$, $s = 1, \dots, k/2$, i.e. $E_s = \{e_{s,1}, \dots, e_{s,k/2}\}$, $s = 1, \dots, k/2$.
- the induced subgraph $G[X \cup Y]$ is the bipartite graph with parts X and Y and with the set of edges E_X , where $E_X = \{(x_{s,1}, y_{s,1}), \dots, (x_{s,k/2}, y_{s,k/2}) \mid s = 1, \dots, k/2, j = 1, \dots, k/2, i = 1, \dots, k/2\}$.
- the induced subgraph $G[Y \cup Z]$ is the union of the k graphs $G[Y_s \cup Z_s]$ which are the complete bipartite graphs $K_{k/2, k/2}$ with parts $Y_s, Z_s, s = 1, \dots, k/2$.
- the induced subgraph $G[Z \cup E]$ is the bipartite graph with parts Z and E and with the set of edges E_Z , where $E_Z = \{(z_{s,1}, e_{s,1}), \dots, (z_{s,k/2}, e_{s,k/2}) \mid s = 1, \dots, k/2, j = 1, \dots, k/2, i = 1, \dots, k/2\}$.
- the induced subgraphs $G[X \cup E]$, $G[Y \cup E]$, $G[Z \cup E]$ are the empty graphs.

The set X is the Core layer, the set Y is the Aggregation layer, the set Z is the Edge layer, and the set E is the Host layer. Notice that the degree of each vertex from $X \cup Y \cup Z$ in G is equal to k .

2 Clos Networks

By a **Clos Network** we mean a:

- $|X| = n_1, |Y| = n_2, |Z| = n_3$.
- the induced subgraph G_X .
- the induced subgraph G_Y .
- the induced subgraph G_Z .

The set X is the Core layer.

3 Fat Tree Network

Note we recall an explanation of layers of $k/2$ switches. Each $k/2$ -partite graph G is a fat tree network if:

- the induced subgraph $G[X \cup Y]$ is a connected graph.
- the set Z can be partitioned into subsets $Z_1, \dots, Z_k, k \geq 1$.
- either the induced subgraph $G[Z_s]$ is a single path with the ends $z_{s,1}, z_{s,2} \in Z_s$ and the sets Z_s and $V \setminus Z_s$ are connected in G only by $\{z_{s,1}, z_{s,2}\}$ and $\{z_{s,2}, z_{s,1}\}$ for some distinct adjacent vertices $z_{s,1}, z_{s,2} \in V$ in the induced subgraph $G[Z_s]$ is a tree with a root $z_{s,1} \in Z_s$ (or with roots $z_{s,1}, z_{s,2} \in Z_s$) and the sets Z_s and $V \setminus Z_s$ are connected in G only by $\{z_{s,1}, z_{s,2}\}$ for some vertex $z_{s,1} \in V$ (or for some distinct adjacent vertices $z_{s,1}, z_{s,2} \in V$); this path or this tree is called a **string**, $s = 1, \dots, k$.
- the set Y is the Aggregation layer (the set of red nodes), the set Z is the Access layer (the set of green nodes).

4 IPBAN Networks

Note we recall an explanation of an IPBAN Network from [4]. The topology consists of two parts: the Access layer and the Aggregation layer. The red nodes form the Aggregation layer while the green nodes form the Access layer. If the total number of the nodes (or servers) in an IPBAN Network is N , the number of aggregation nodes is about $0.1N$, while the rest $0.9N$ nodes belong to the access layer. For each pair of the aggregation nodes, there are at most 20 rings, and each ring consists of about 20 access nodes (green nodes). A basic principle for the IPBAN is the traffic should not be transmitted by the access rings. So the link cost in the access ring is larger than the aggregation links. Without loss of generality, the link cost can be set as follows, the link in the access ring is set to 100, while 10 in the aggregation network.

происходит группировка

атрибутируемых топологий?

2 Problem statement for Indirection Next Hop

The goal is to devise an algorithm A with inputs G, g_n, r, B, H, D and outputs G', H', H', H' . Inputs are as follows: a graph $G \in \mathcal{G}$; a description $g_n \in \mathcal{M}$ of changes from a predefined class of possible graph mutations $\mathcal{M} \subseteq \mathcal{G}^2$ (for instance addition or deletion of a single vertex in graph) that applied to G results in a graph $G' \in \mathcal{G}$; a function $r: V \rightarrow V$ representing a routing function; a function $B: V \rightarrow V$ representing a function that includes information about G' useful for the next run.

The informal meaning of G' is to represent a view on a network, where vertices are standalone agents (computers, routers, etc.), vertices show their immediate connections, and weight function represents "costs" of transferring message with a also shown in G' and H' , who maintain a \mathcal{M} message context data $\{DE\}_{t \in T}$, organizing and modifying it.

We define the graph $R_k \subseteq G$ of all nodes to d originating from vertices of V_k by following:

$$R_k = \{v \in V \mid \exists p \in P_k \text{ such that } (v, p) \in E \wedge (v, p) \in R_k\}$$
$$R_k = \{v \in V \mid \exists p \in P_k \text{ such that } (v, p) \in E \wedge (v, p) \in R_k\}$$

A network state N associated with G is called correct when

$$\forall v \in V \{ (v \in N) \wedge (\exists p \in P_k \text{ such that } (v, p) \in E \wedge (v, p) \in R_k) \}$$

When all induced vertices states of N are correct the following statements are equivalent for all $d \in V_k$:

- R_k has no cycles or $\{v \in V_k \mid \exists p \in P_k \text{ such that } (v, p) \in E \wedge (v, p) \in R_k\}$ is a directed acyclic graph.
- Let us now consider a change in the network topology from a graph G to a graph G' . We say that a vertex state $v = (G', r, B, H, D)$ follows a vertex state $v = (G, r, B, H, D)$ and write $v \in F$ if $G' \subseteq G$ and $(G' \setminus G) \cap A \subseteq \emptyset$, where $A = \{(v, w) \in E \mid (v, w) \in G' \wedge (v, w) \notin G\}$.
- We say that a network state $N = \{N_i\}_{i \in I}$, $N_i = (G_i, r_i, B_i, H_i, D_i)$ for $i \in I$, follows a network state $N = \{N_i\}_{i \in I}$, $N_i = (G_i, r_i, B_i, H_i, D_i)$ for $i \in I$, and write $N \in \mathcal{N}$ if $\forall i \in I \{ N_i \in F \}$.
- $\forall v \in V_k \{ (v \in N) \wedge (\exists p \in P_k \text{ such that } (v, p) \in E \wedge (v, p) \in R_k) \}$.

An algorithm A is called correct if for all network states N, N' , $(N \in \mathcal{N}) \wedge (N' \in \mathcal{N}) \Rightarrow (N \in \mathcal{N})$ is correct. The goal is to construct a correct algorithm A subject to the following quality criteria.

We compare the following parameters: the computational complexity of the algorithm; the space requirements for service data B ; the n -length of (v, w) paths in R_k ; the number of domains in partition R_k ; the dissimilarity between partitions R_k and R_l .

Now we give an explanation of the Indirection Next Hop problem statement. This problem is considered for a given graph $G = (V, E)$. For each vertex $v \in V$ we need to find a partition R_k of the set $V \setminus \{v\}$ and a function $H_k: R_k \rightarrow E$ for which the following two properties hold:

- If $W \subseteq V \setminus \{v\}$, $W \in R_k$, then $H_k(W) = (v, w)$ for a certain $w \in V$. This means that the edge $H_k(W)$ starts at the vertex v . Intuitively, this edge $H_k(W)$ is the next hop edge for d from v .
- If $d \in V \setminus \{v\}$, then there exists an H_k -path P in G such that $P = (v, v_1, v_2, \dots, v_{k-1}, v_k = d)$, where $v_i \in V \setminus \{v\}$, $v_i \in R_{k-1}$, then $H_{k-1}(v_i) = (v_i, v_{i+1})$. This means that, for any vertex $v \in V \setminus \{v\}$, there exists a path P from v to d in G , and this path P consists only of the next-hop edges for d .

In addition, a weight function $w: E \rightarrow \mathbb{R}^+$ is given for the graph $G = (V, E)$. We evaluate the w -length $w(P) = \sum_{(v, w) \in P} w(v, w)$ for the next-hop path P ; this weight length represents the costs of transferring a message from v to d in the network.

Finally, the graph $G = (V, E)$ can be changed by two ways: an existing edge $e \in E$ can be deleted from the graph and a new edge $e \notin E$ can be added into the graph. This corresponds to modifications of the network, for example, old links can disappear and new links can appear in the network. If the graph is changed then we need to reconstruct the partitions R_k and the functions H_k to find a new partition R'_k and a new function H'_k . We evaluate the dissimilarity between partitions R_k and R'_k ; this dissimilarity shows the expense to recompute the next-hop tables.

Обзор схожих работ



- The problem statement similarity
- The applicability to the given topologies
- The algorithm complexity
- The accuracy of the solution

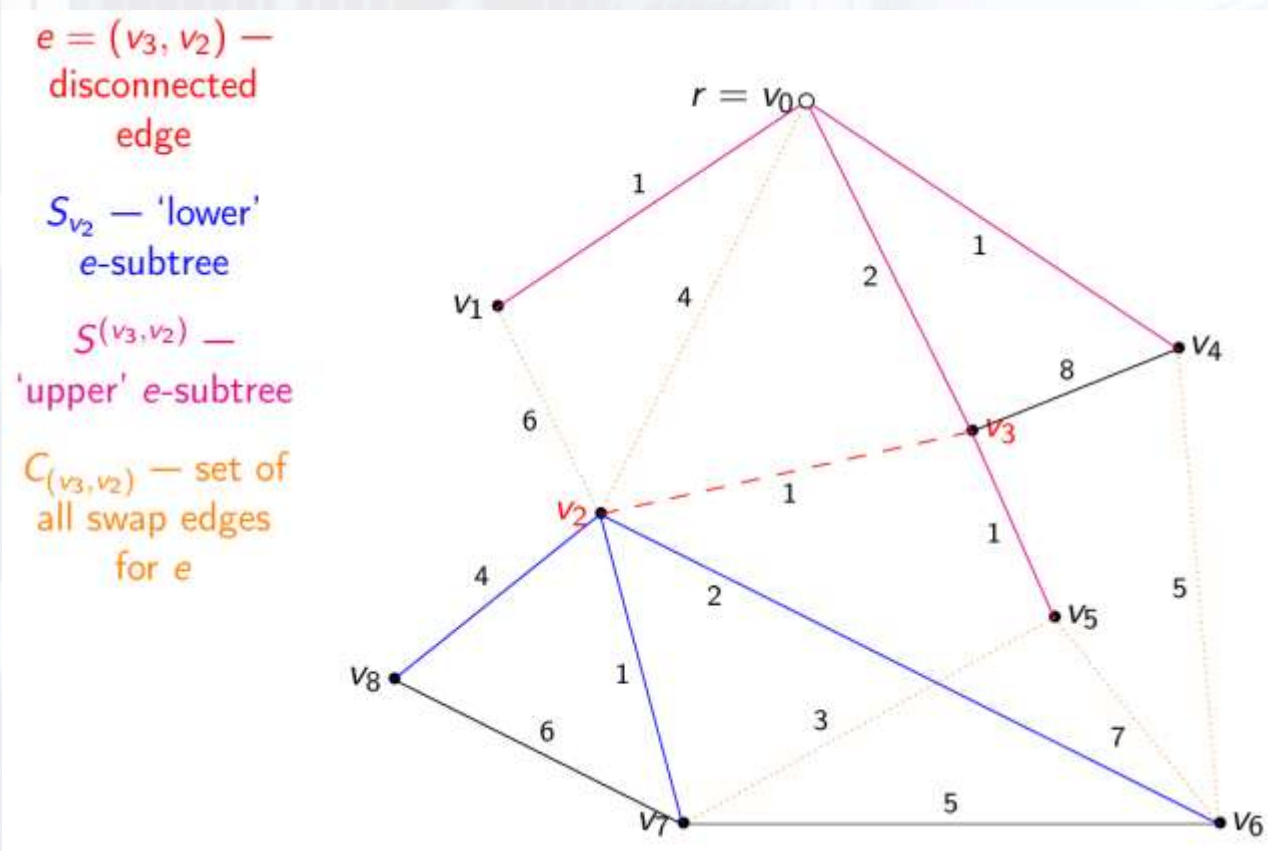
The correspondence of articles to selected overview criteria

Article	The problem statement similarity	The applicability to the given topologies	The algorithm complexity	The accuracy of the solution
Ramalingam G., Reps T., On the computational complexity of dynamic graph problems (1996) [6]	+	+	+	+
Frigioni D., Marchetti-Spaccamela A., Nanni U. Fully dynamic output bounded single source shortest path problem (1996) [7]	+	+	+	+
Bin Xiao, Jiannong Cao, Zili Shao, Edwin H.-M. Sha. An efficient algorithm for dynamic shortest path tree update in network routing (2007) [8]	+	+	+	+
Khuller S., Raghavachari B., Young N., Balancing minimum spanning trees and shortest-path trees (1995) [9]	-	+	+	+
Nardelli E., Proietti G., Widmayer P., How to swap a failing edge of a single source shortest paths tree (1999) [4]	+	+	+	-
Nardelli E., Proietti G., Widmayer P., Swapping a failing edge of a single source shortest paths tree is good and fast (2003) [5]	+	+	+	-
Di Salvo A., Proietti G. Swapping a failing edge of a shortest paths tree by minimizing the average stretch factor (2007) [10]	+	+	+	-
Spira P. M., Pan A., On finding and updating spanning trees and shortest paths (1975) [11]	+	+	+	-
Roditty L., Zwick U., On dynamic shortest paths problems (2004) [12]	+	+	-	+
Williams V. V., Faster replacement paths (2011) [13]	-	+	-	+
Guo Y., Qin Z., Chang Y., A novel hybrid algorithm for the dynamic shortest path problem (2010) [14]	+	+	+	-
Cong J., Kahng A. B., Robins G., Sarrafzadeh M., Wong C. K., Provably good performance-driven global routing (1992) [15]	-	+	+	-
Eppstein D., Italiano G. F., Tamassia R., Tarjan R. E., Westbrook J., Yung M., Maintenance of a minimum spanning forest in a dynamic plane graph (1992) [16]	-	+	+	-
Nagamochi H., Ibaraki T., A linear-time algorithm for finding a sparse k -connected spanning subgraph of a k -connected graph (1992) [17]	-	+	+	+

Разработка алгоритма



- Swap-edge based algorithm



- Dynamic algorithm

Реализация алгоритма



Структуры данных

1. Матрица смежности
2. Список смежности (список)
3. Список смежности (хэш-таблица)
4. Список смежности (дерево поиска - будут приведены значения для AVL, но можно использовать и любое другое)

Структура	Место	get e	add e	del e	add v	del v
Mc	$O(V^2)$ (value/8 if no weight)	$O(1)$	$O(1)$	$O(1)$	$O(V) * \max(O(\text{array copy}), O(\text{array alloc}))$	$O(1)$ if do not care about space, $O(V) * \max(O(\text{array copy}), O(\text{array alloc}))$ if reallocating
Cc(c)	$O(E+V)$	$O(V)$	$O(1)$	$O(V)$	$O(1)$	$O(1)$
Cc(x)	$O(E+V)$ (const worse than list)	$O(1) *$ $O(\text{hash})$	$O(1) *$ $O(\text{hash})$	$O(1) *$ $O(\text{hash})$	$O(1)$	$O(1)$
Cc(d)	$O(E+V)$ (const worse than list)	$O(\log V)$	$O(\log V)$	$O(\log V)$	$O(1)$	$O(1)$

Экспериментальное исследование



Эффективность:

$$\frac{\sum_i Time_{nh,i} - Time_{def,i}}{|V|}$$

$$\frac{\sum_{i,j} d_{G_{nh}}(v_i, v_j) - \sum_{i,j} d_{G_{def}}(v_i, v_j)}{|V|}$$

Параметры эксперимента:

- Topology family: Clos or IPRAN.
- Overall number of vertices (routers) in topology graph. Possible values: 500,1000,2500,5000,6000,7000,8000,9000,10000
- Distribution of subnetworks that defines initial routing table:
 - Static: 5,10,15 subnetworks per vertex.
 - Uniform distribution $U(1,20)$.
- Sequence of graph mutations gm:
 - Trivial sequences of length 1.
 - Sequences where after several removal of edge/vertex follows addition of same edge/vertex.
- Graph structure used in algorithm implementation:
 - Adjacency list with edges stored in list.
 - Adjacency list with edges stored in array.
 - Adjacency list with edges stored in set based on hash table.
 - Adjacency list with edges stored in set based on red-black tree.
- Used algorithm: Dijkstra, proposed dynamic Dijkstra, proposed swap-edge based algorithm

ML в балансировке трафика



$$G(t_i) = (V(t_i), E(t_i))$$

- $c_{u,v}$ - the bandwidth of the corresponding link; assume $c_{u,v} = c_{v,u}$.
- $b_{u,v}$ - the currently occupied bandwidth of the link. Note, $b_{u,v}$ and $b_{v,u}$ can differ from each other.

$$PATHS(u, v) = \{(d_i, w_i)\}, \forall u, v \in V, u \neq v$$

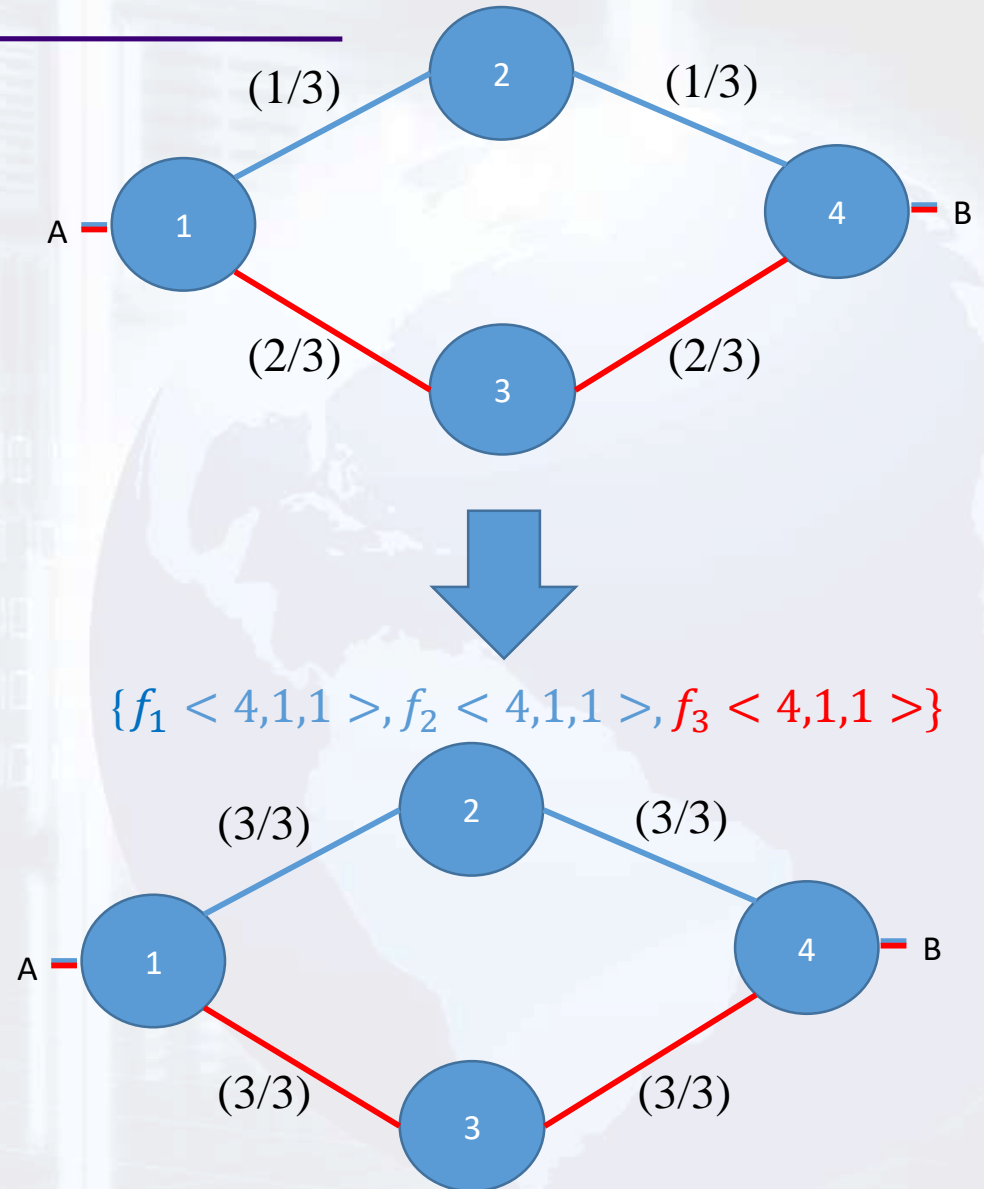
- IGP-path from vertex k to vertex v - (d, w)
- $PATHS(1,4) = \{(2, 2), (3, 2)\}$

TM

- Flow is $\langle f_{dst}, f_{src}, fr \rangle$
- $TM = \{f_1 \langle 4, 1, 1 \rangle, f_2 \langle 4, 1, 1 \rangle, f_3 \langle 4, 1, 1 \rangle\}$

$$HASH: G(t_i) \times R(t_i) \times TM(t_{i+1}) \rightarrow G(t_{i+1})$$

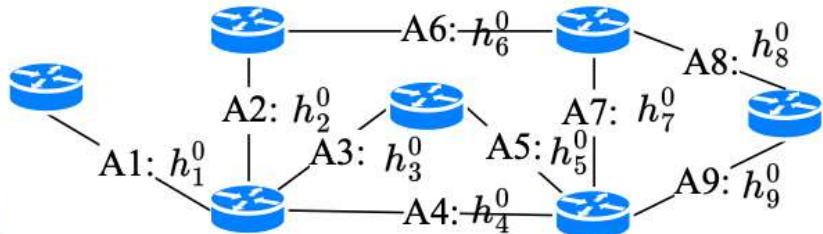
- $R(t_i)$ - hash-weights (h -weights)
- $R = \{2, 1\}$



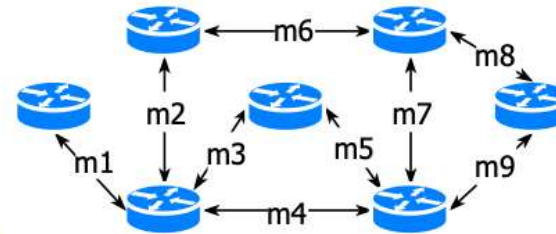
MARL-GNN подход



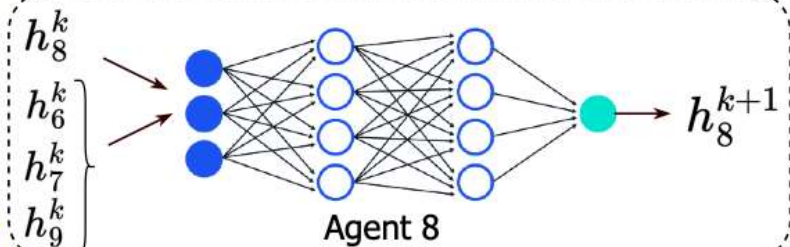
(a) Каждый агент инициализирует свой вектор состояния (hidden state), используя некоторые начальные признаки, уже включенные во входной граф



(b) Агенты обмениваются своими текущими векторами состояний с соседними агентами

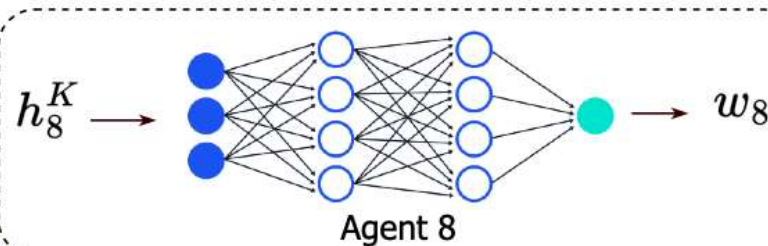


(c) Агенты обрабатывают полученные сообщения и обновляют свои векторы состояний на основе агрегированной информации



к
итераций

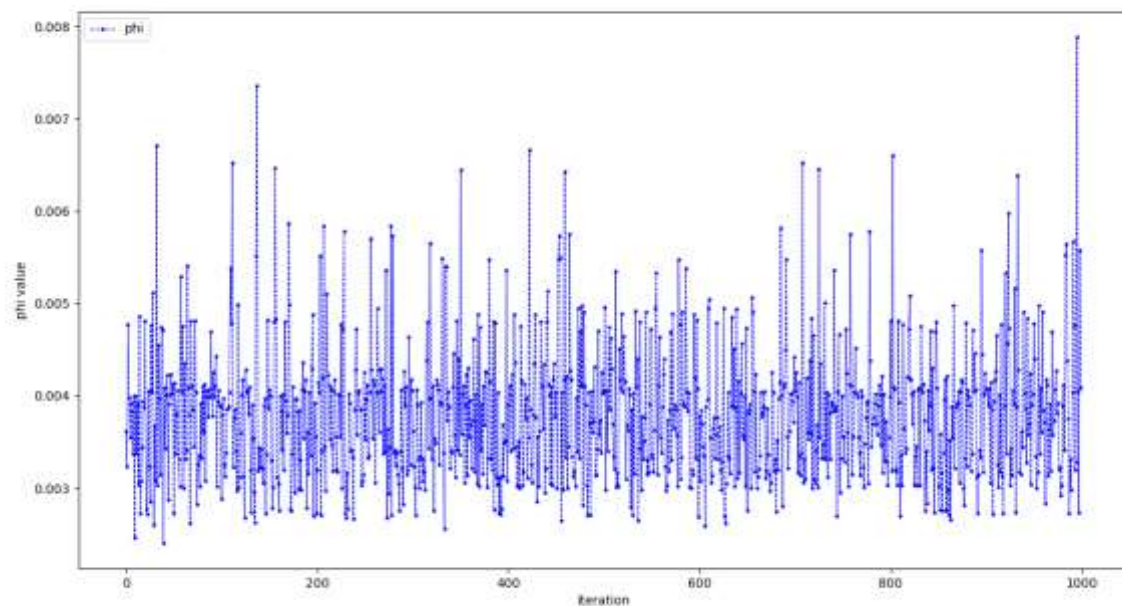
(d) Агенты совершают действие (т. е. изменяют вес линка) на основе полученного вектора состояния, а среда обновляет веса ребер в конфигурации графа



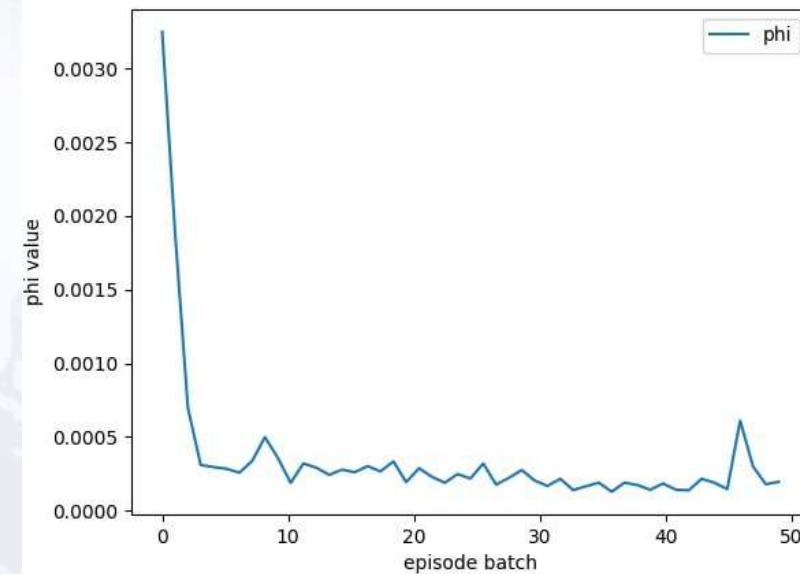
Экспериментальное исследование



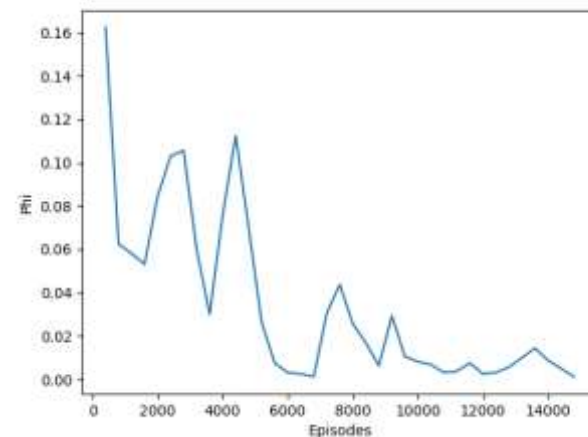
Проблема с осцилляцией:



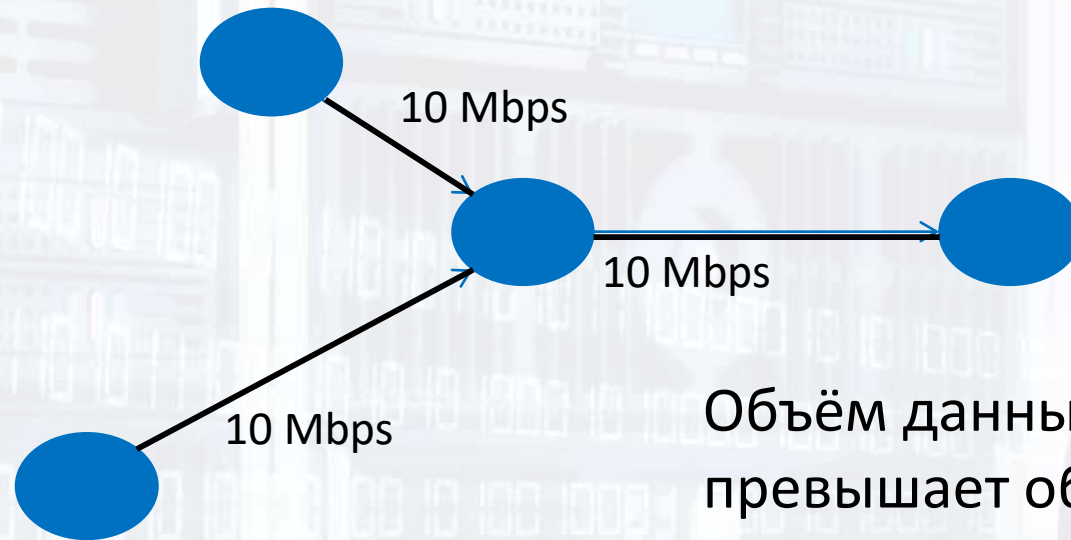
2 flows



50 flows



Перегрузка

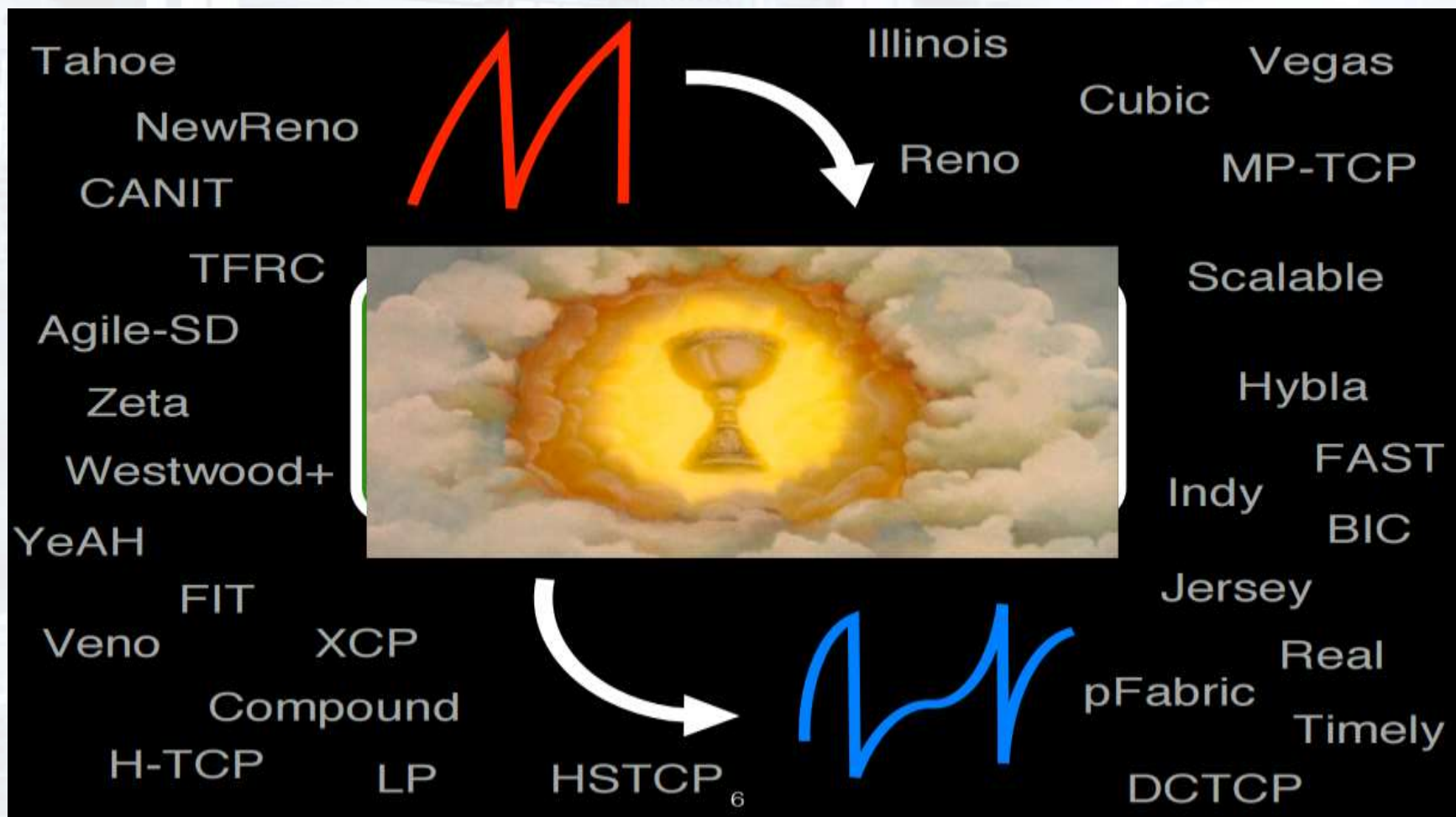


Объём данных, поступающих на коммутатор, превышает объём данных, которые он может передать

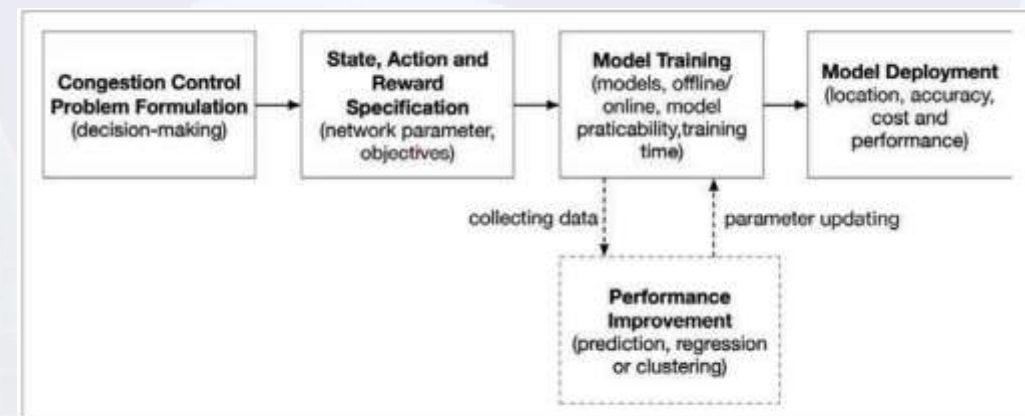
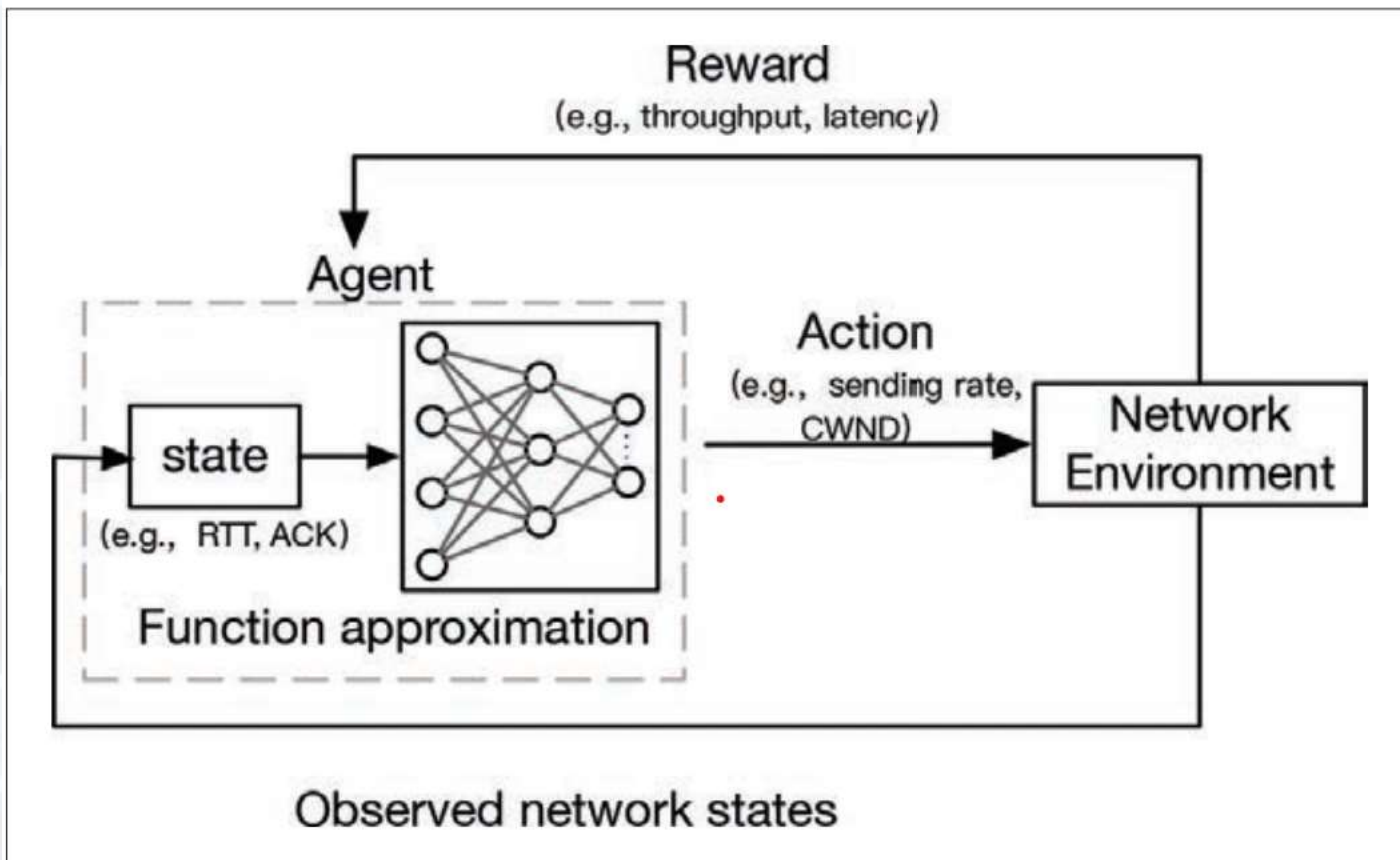
Коммутаторы буферизируют проходящий через них трафик:

- При кратковременной перегрузке увеличивается задержка
- При достаточно долгой перегрузке пакеты сбрасываются

Алгоритмы управления перегрузкой



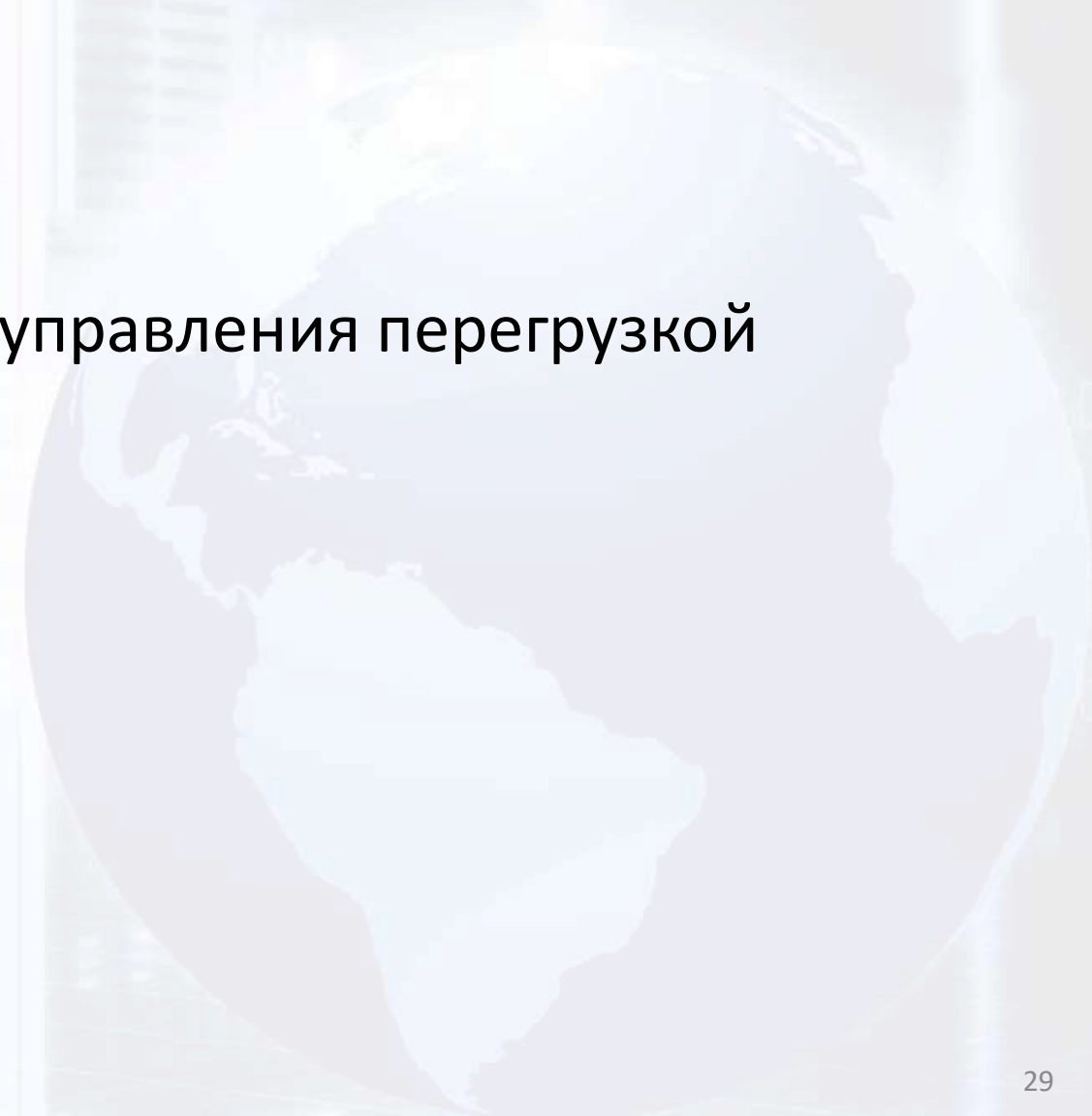
ML в алгоритмах управления перегрузкой



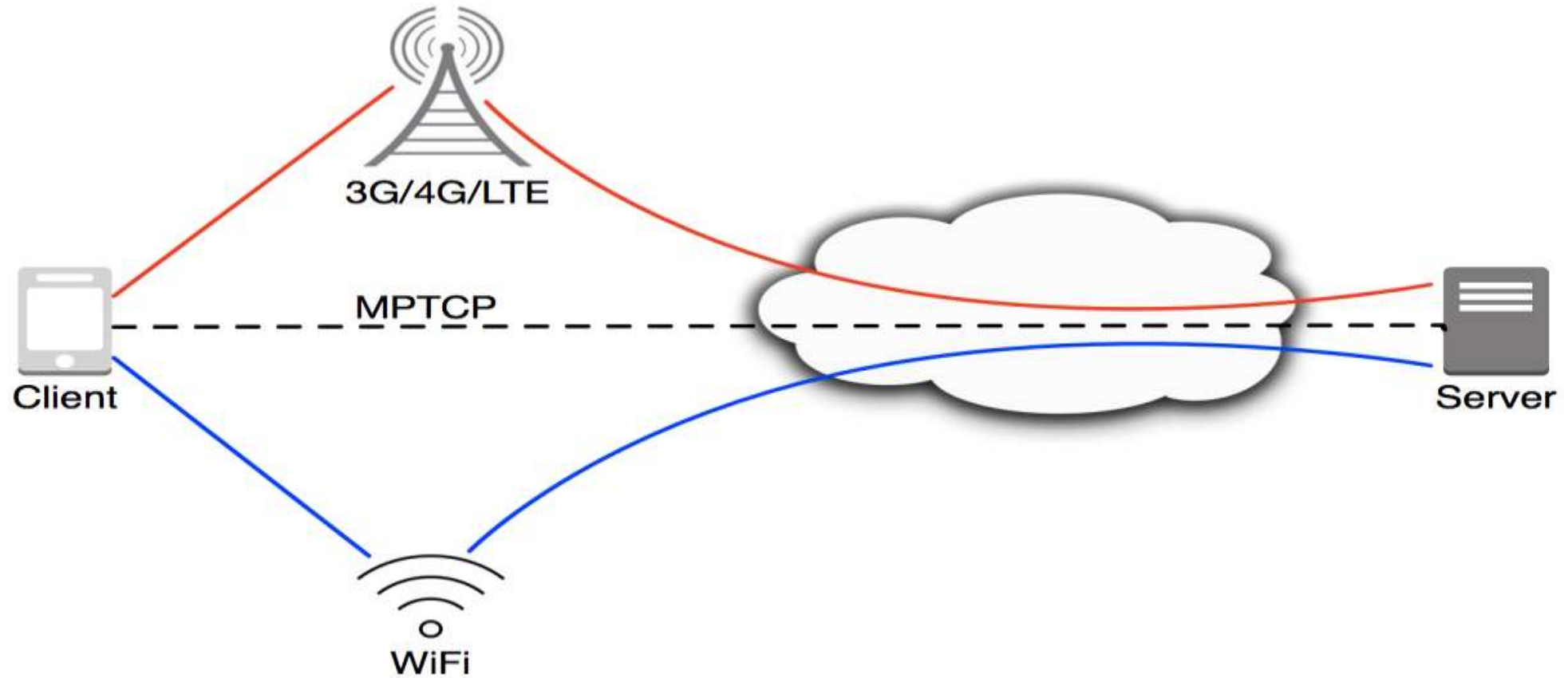
Технологии и задачи



- Машинное обучение
- Оптимизация параметров
- Разработка и анализ новых алгоритмов управления перегрузкой
 - BBR, RCP



Demultiplexed Transport Connections



Demultiplexed Transport Connections (FDMP)



Application Layer



Стандартный socket API

Адаптивное управление потоками

Планирование пакетов

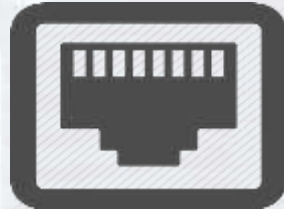
Transport Layer

Подпоток

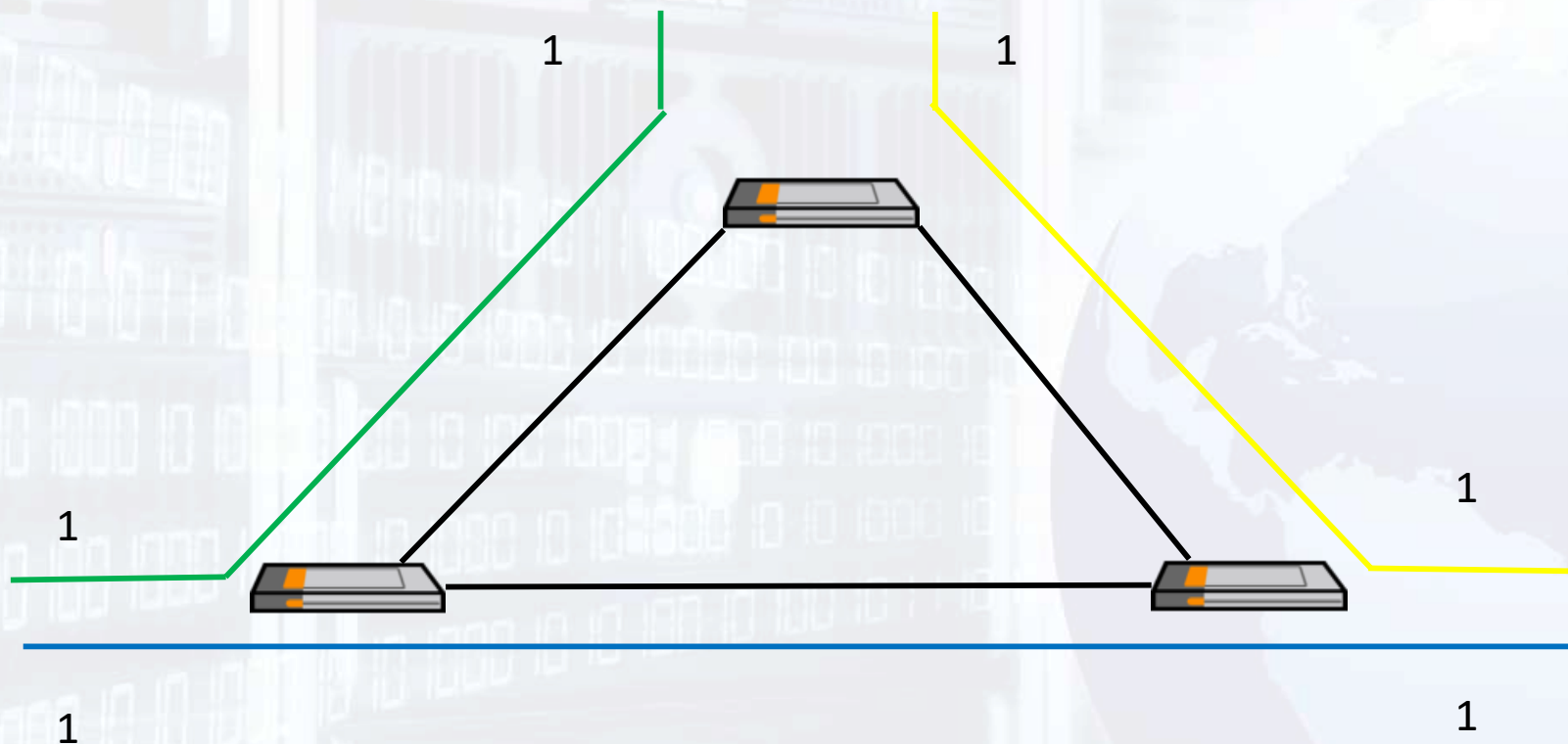
Подпоток

Подпоток

Network Layer

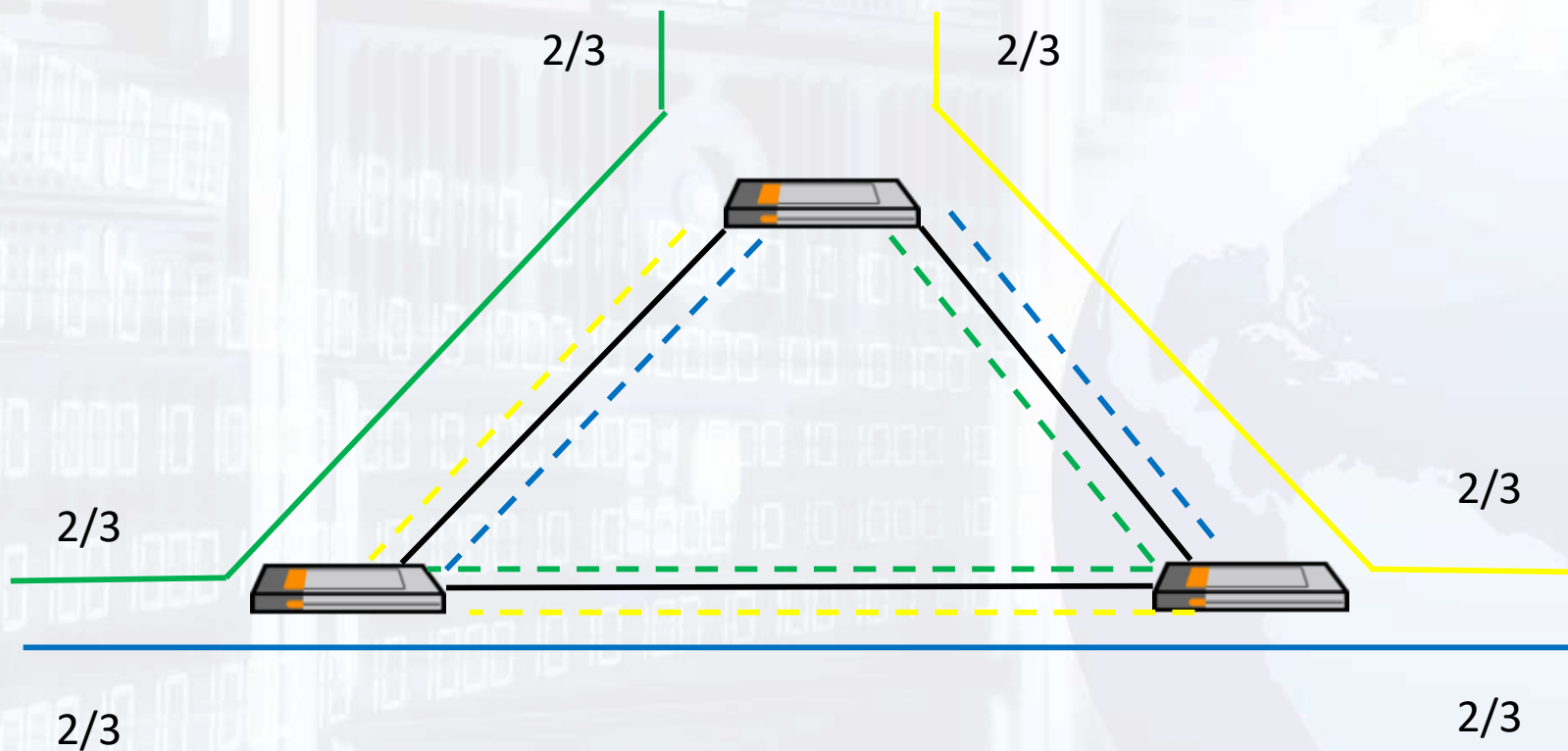


Массовая многопоточность



Общая пропускная способность сети - 6

Массовая многопоточность

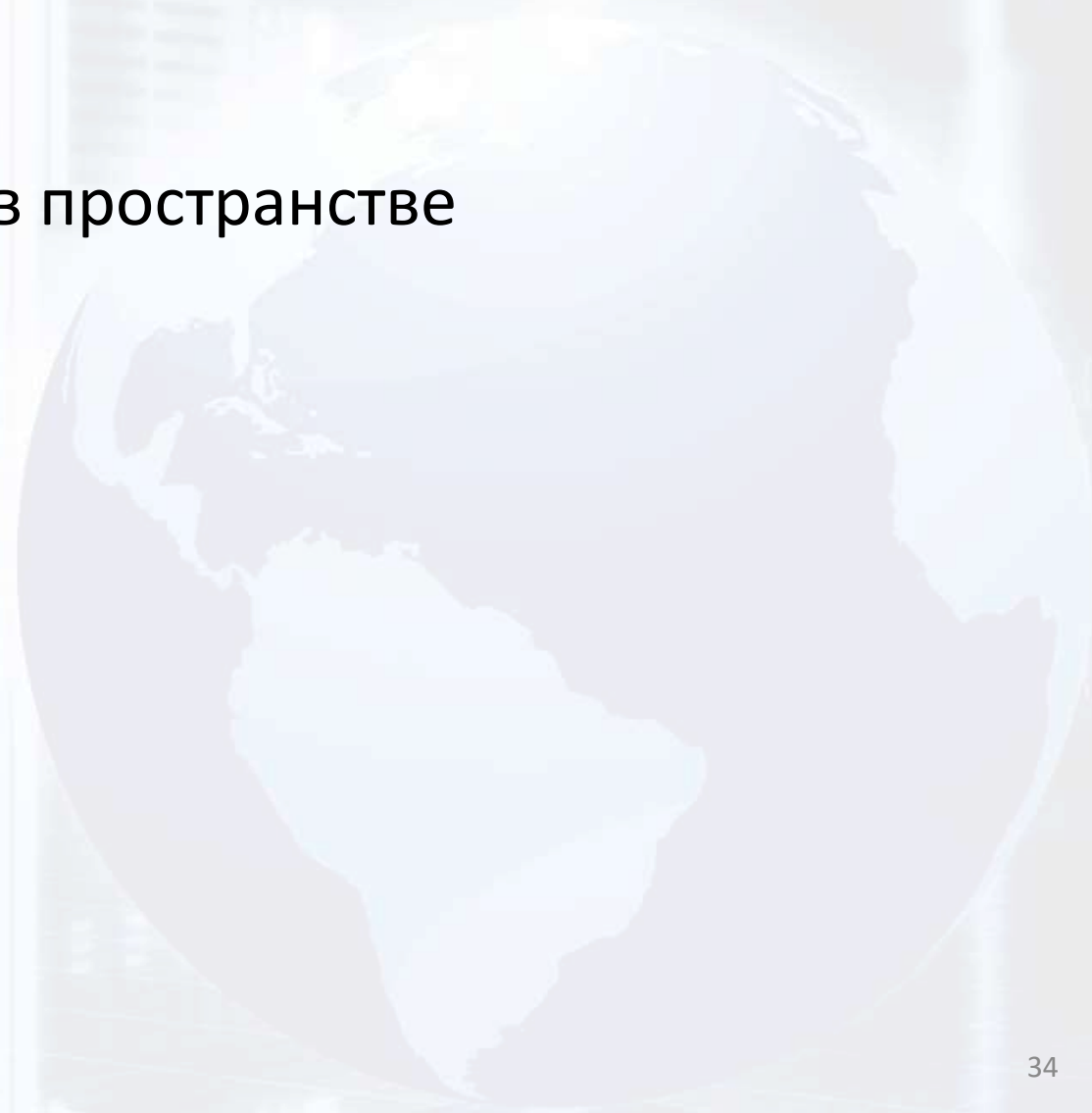


Общая пропускная способность сети - 4

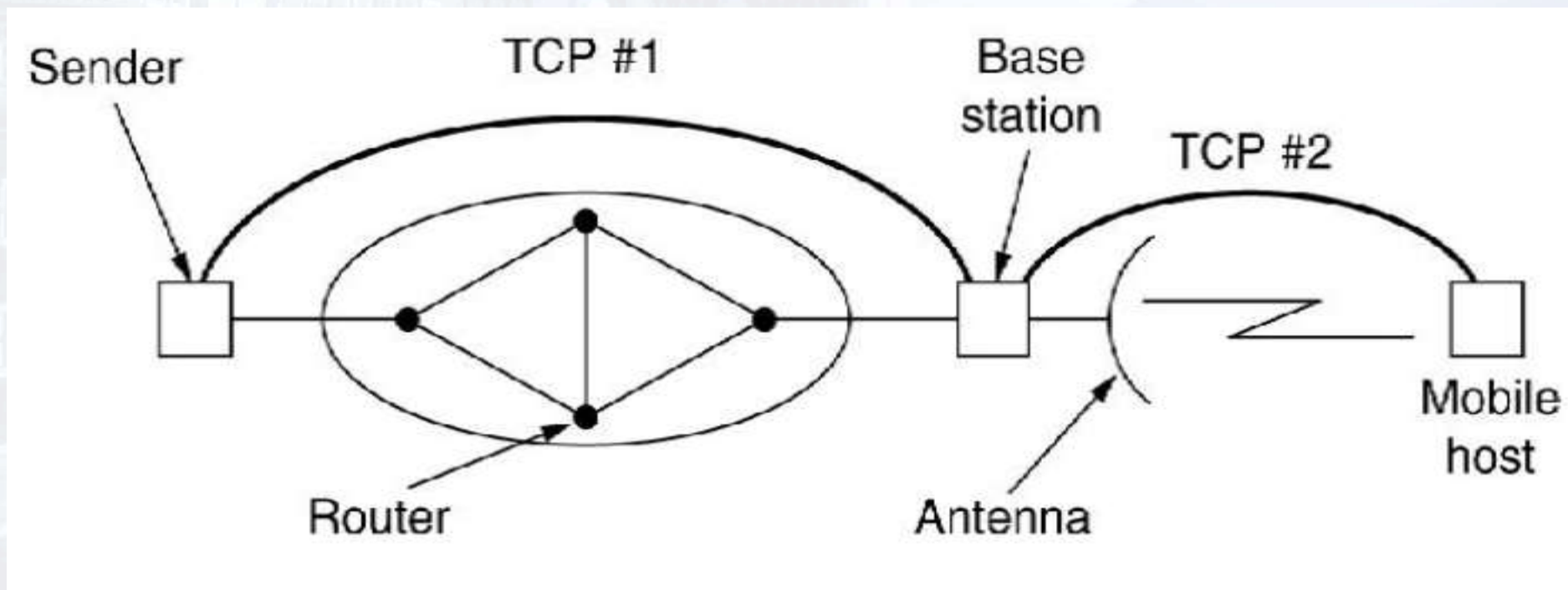
Технологии и задачи



- Возможность работы с ядром Linux
- Оптимизация работы с сетевым стеком в пространстве пользователя (dpdk)
- Массовая многопоточность
- Обеспечение качества сервиса
- Умная маршрутизация
- Алгоритм управления перегрузкой



Сегментирование - Split TCP



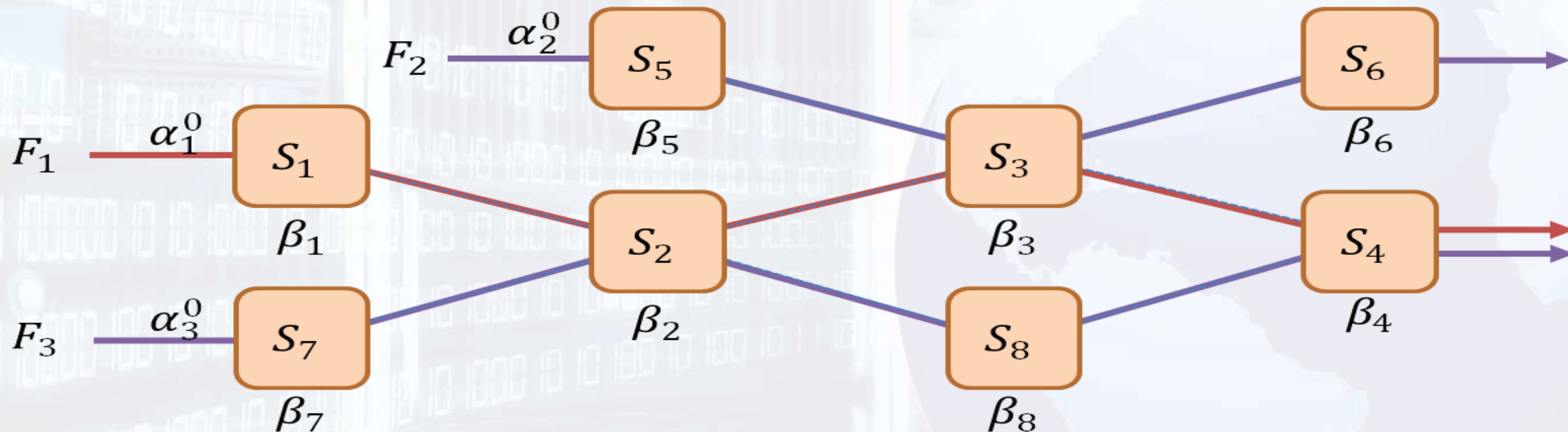
Технологии и задачи



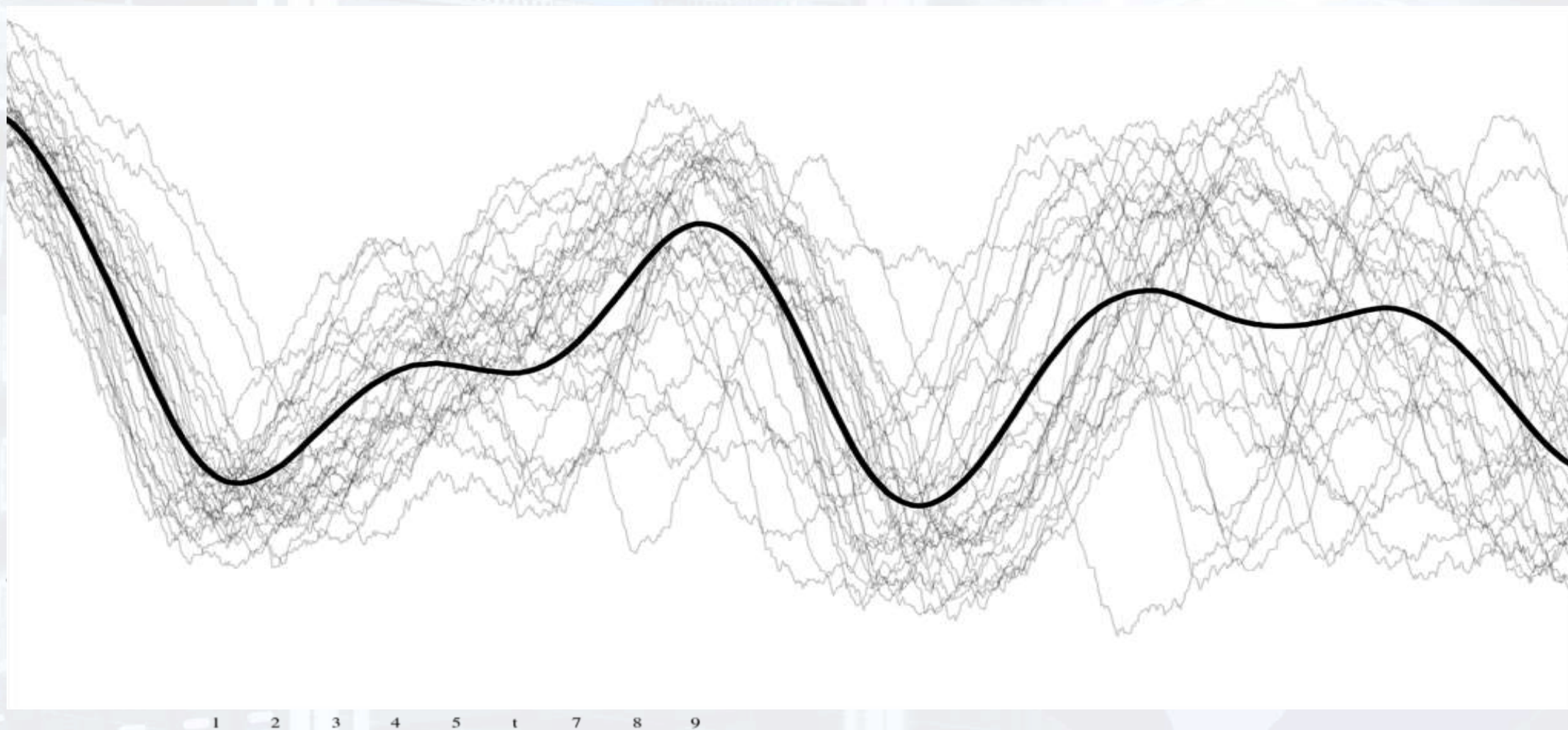
- Разработка прокси для split TCP
- Размещение прокси в сети
- Маршрутизация в такой сети
- Оценка метрик качества сервиса для split-соединений



Сетевое исчисление



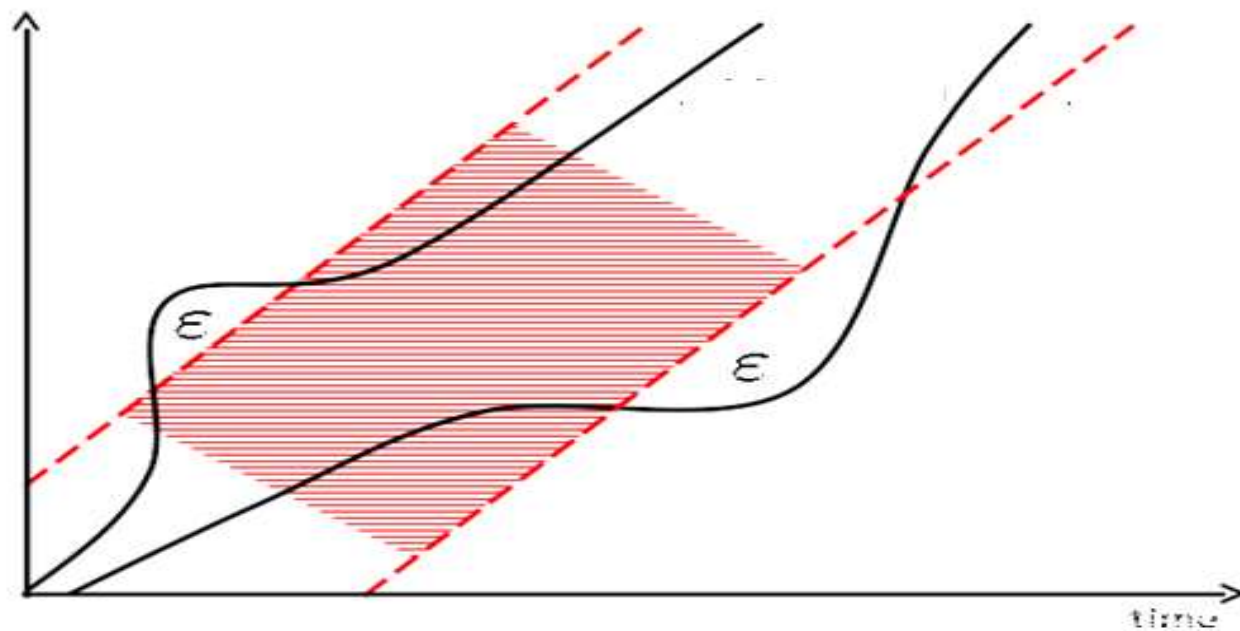
Сетевое исчисление



Стохастическое сетевое исчисление



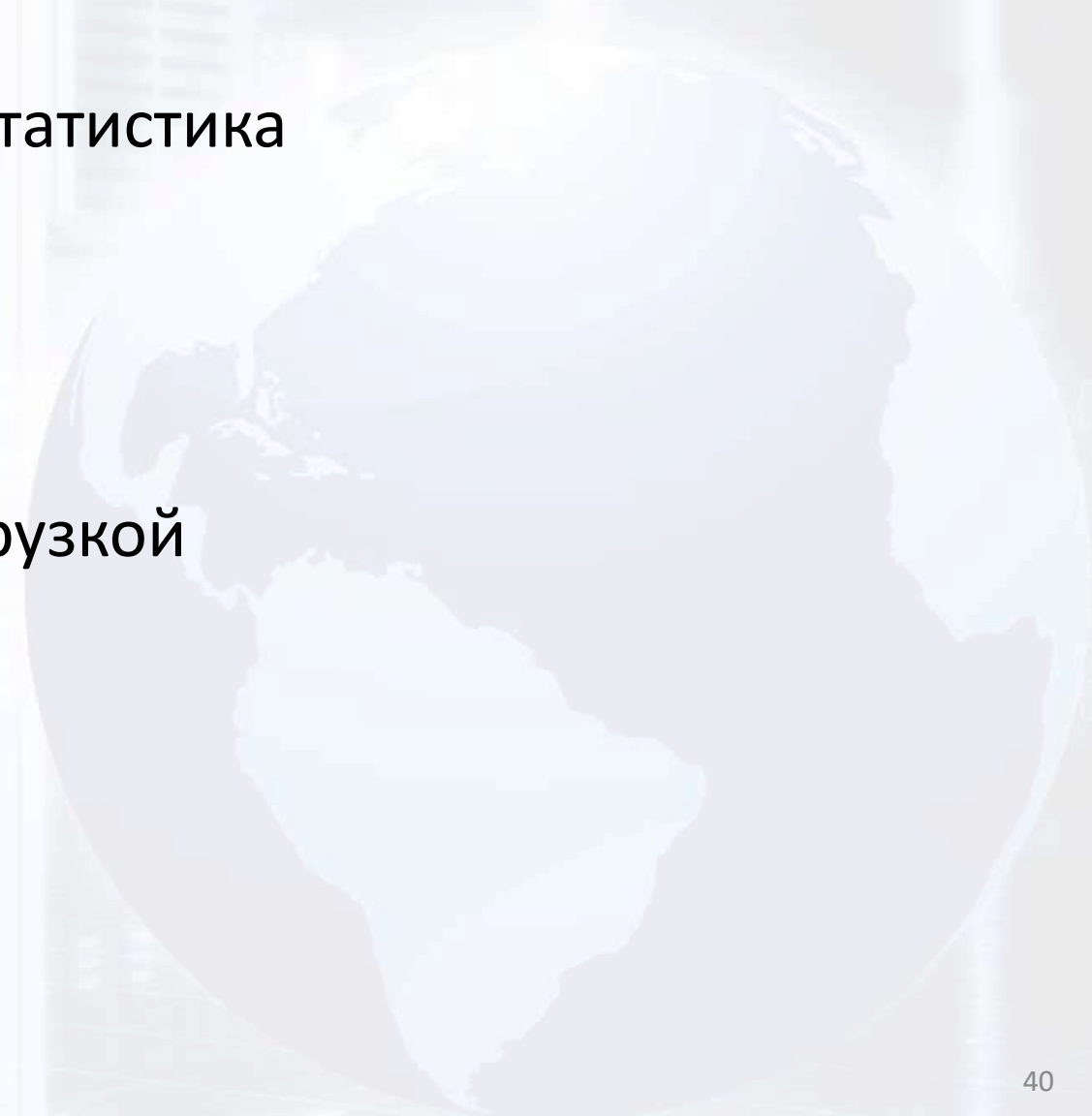
Позволить нарушения кривых нагрузки и сервиса с некоторой вероятностью
-> теряем точность результата



Технологии и задачи



- Теория вероятностей, математическая статистика
- Линейное программирование
- Оптимизация параметров
- Учет потери пакетов
- Связь с алгоритмами управления перегрузкой





Благодарю за внимание!