

# ДИНАМИЧЕСКАЯ МИГРАЦИЯ СЕРВИСОВ БПЛА МЕЖДУ EDGE-УЗЛАМИ С ОБЕСПЕЧЕНИЕМ ВЫСОКОЙ ДОСТУПНОСТИ

Пигин С.В., Акбаров О.И.

Московский авиационный институт (национальный исследовательский университет),  
Москва, Россия

psv\_1961@mail.ru, iserver12345@gmail.com

*Аннотация. Беспилотным летательным аппаратам придается особое значение в развитии экономики Российской Федерации с учётом условий выполняемых задач. Данное исследование посвящено динамической миграции сервисов управления изделиями между пограничными узлами для обеспечения высокой доступности. Предложено применение контейнерных платформ для минимизации времени простоя.*

*Ключевые слова: беспилотные летательные аппараты, edge-computing, высокая доступность, миграция сервисов, Kubernetes, отказоустойчивость.*

## Введение

Беспилотные летательные аппараты (БПЛА) стали неотъемлемым инструментом в различных сферах деятельности. Успешность их применения напрямую зависит от бесперебойности систем управления и обработки данных.

Для удовлетворения требований к задержкам и объёму обработки данных в реальном времени парадигма edge-computing использует перенос вычислительных ресурсов на периферию сети, ближе к источникам данных – наземным станциям управления и маршрутам полета БПЛА. Однако физическое перемещение дронов, перегрузка или отказ отдельных edge-узлов, колебания качества связи могут сделать текущий узел размещения сервиса управления или обработки телеметрии недоступным или неэффективным.

Для решения этих проблем и достижения высокой доступности (High Availability, HA) используется динамическая миграция сервисов между географически распределёнными edge-узлами. Её суть заключается в автоматизированном и оперативном переносе контейнеризованных сервисов БПЛА на оптимально расположенный узел при изменении условий окружения (покрытие сети, нагрузка, отказ). Построение отказоустойчивых edge-инфраструктур, способных поддерживать динамическую миграцию, немыслимо без современных платформ контейнерной оркестрации. Kubernetes (K8s) является стандартом в этой области, хотя его настройка для edge-сценариев с гарантиями HA требует проверки. В рамках работы использован инструментальный MBRC (MasterBpro Ready Cluster) [1, 2].

Целью данного исследования является систематизированный обзор современных методов и технологий, обеспечивающих динамическую миграцию сервисов БПЛА между edge-узлами с соблюдением требований высокой доступности. Для достижения этого работа включает анализ существующих архитектурных решений и технологий HA в распределённых edge-инфраструктурах, детальное описание типовых сценариев миграции сервисов управления БПЛА (таких как перемещение между зонами покрытия, отказ узла, балансировка нагрузки), исследование методов обеспечения отказоустойчивости и минимизации времени простоя при миграции контейнеризованных сервисов.

Ключевые слова: беспилотные летательные аппараты, edge-computing, высокая доступность, миграция сервисов, Kubernetes, CRJU, отказоустойчивость

## 1. Архитектура и технологии обеспечения высокой доступности сервисов БПЛА в edge-средах

Распределённая модель вычислений, известная как Edge-computing, предполагает обработку данных максимально близко к их источнику [3, 4], например, БПЛА или сенсорам, что критически важно для снижения задержек и разгрузки центральных ЦОД. Неотъемлемой частью надёжной работы таких систем является High Availability (HA) – способность обеспечивать непрерывное функционирование даже при сбоях компонентов, достигаемая за счёт резервирования, автоматического перераспределения нагрузки и быстрого восстановления сервисов.

Обеспечение HA в распределённых edge-инфраструктурах базируется на нескольких фундаментальных принципах. Ключевым из них является избыточное развертывание сервисов, подразумевающее запуск множественных экземпляров контейнеров на разных физических узлах. Не менее важна оркестрация и отказоустойчивость, обеспечивающие автоматический перезапуск и перенос сервисов в случае выхода узла из строя. Для сохранения целостности системы также

необходима репликация состояний, гарантирующая синхронизацию критически важных данных, таких как телеметрия и сессии управления, между всеми задействованными узлами. Реализация этих принципов на практике опирается на ряд специализированных технологий.

Kubernetes (рис. 1) служит базовой платформой для оркестрации контейнеров, предоставляя такие механизмы, как ReplicaSets, PodDisruptionBudget и аннотации для тонкого управления стратегиями миграции. Для специфики edge-среды часто используется KubeEdge – надстройка над Kubernetes, создающая двунаправленный туннель управления и синхронизирующая состояние между облачным ядром и периферийными устройствами. Консистентность конфигурации и состояния кластера обеспечивается etcd-высокодоступным распределенным хранилищем типа «ключ-значение».

## Что такое Kubernetes?

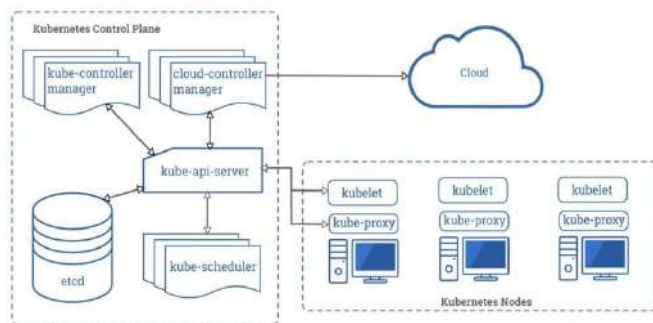


Рис. 1. Архитектура кластера [4]

Управление сетевым взаимодействием, мониторинг и контроль трафика между микросервисами берут на себя решения для сервисной сетки (service mesh), такие как Linkerd (рис. 2) или Istio. Для минимизации времени простоя при переносе контейнеров применяется CRIU (Checkpoint/Restore In Userspace) [4], позволяющий «замораживать» состояние работающего процесса и восстанавливать его на другом узле.

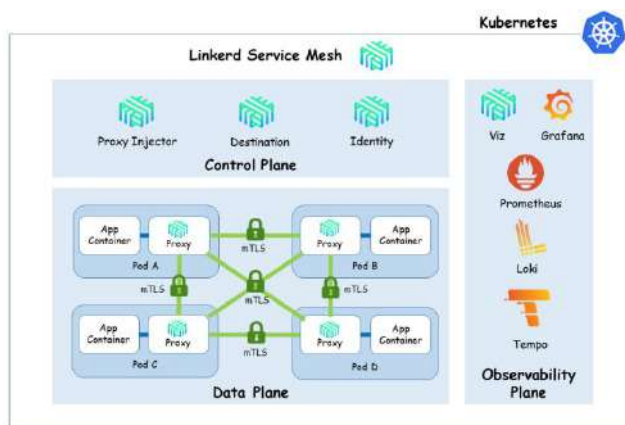


Рис. 2. Пример реализации на базе сервисной сетки [4]

Помимо инструментов уровня оркестрации, существуют готовые коммерческие и открытые платформы, предлагающие комплексные решения для edge. AWS IoT Greengrass расширяет возможности AWS Lambda на периферийные устройства, поддерживая локальные вычисления и синхронную репликацию данных. Azure IoT Edge от Microsoft позволяет оркестрировать модули с использованием Azure Kubernetes Service. Открытая среда OpenNESS (Intel) [5, 6] предоставляет средства для управления жизненным циклом приложений и реализации сетевых функций на edge.

Таким образом, современные подходы к обеспечению высокой доступности в edge-средах сочетают стандартные механизмы Kubernetes с дополнительными инструментами (такими как KubeEdge, сервисные сетки, CRIU) и готовыми коммерческими платформами. Их общая главная задача – гарантировать минимальное время простоя и строгую консистентность состояния во время динамической миграции сервисов между edge-узлами.

Выбор стратегии самой миграции сервисов БПЛА тесно связан с моделью взаимодействия между дронами и edge-инфраструктурой. В централизованной модели все решения о миграции принимает единый контроллер, который, основываясь на мониторинге состояния узлов и БПЛА, выбирает оптимальное место для размещения сервиса. Децентрализованная модель наделяет edge-узлы автономией: они самостоятельно принимают решения о миграции, используя локальные метрики нагрузки и координируясь напрямую друг с другом без центрального управляющего звена. Гибридная модель стремится найти баланс: высокоуровневое управление осуществляется централизованным контроллером, но отдельные решения, требующие быстрой реакции на локальные изменения, могут приниматься непосредственно на edge-узлах [7].

### 1.1. Факторы, влияющие на работу

Динамическая миграция сервисов БПЛА между edge-узлами оказывает комплексное влияние как на производительность системы, так и на ее отказоустойчивость. С одной стороны, она значительно повышает гибкость и эффективность использования ресурсов инфраструктуры, позволяя адаптироваться к изменяющимся условиям. С другой стороны, сам процесс миграции может сопровождаться временным увеличением задержки, особенно в момент передачи состояния сервиса между узлами, что потенциально влияет на время реакции. Что касается отказоустойчивости, то грамотно реализованная динамическая миграция служит ключевым механизмом обеспечения высокой доступности, позволяя минимизировать простои и избежать потери управления дроном. Однако некорректная настройка этого процесса, напротив, способна вызывать нежелательные прерывания связи и потерю критически важной телеметрии с БПЛА.

Миграция сервисов может инициироваться в различных типовых сценариях. Плановая миграция осуществляется в рамках заранее запланированных мероприятий, таких как техническое обслуживание или обновление узлов, когда сервисы перемещаются на резервные ресурсы без спешки. Аварийная миграция, напротив, представляет собой экстренный ответ на нештатную ситуацию — отказ оборудования или критическое падение производительности узла, требующий немедленного переноса сервисов для восстановления работоспособности. Отдельным важным сценарием является миграция по географическому признаку, когда управляющие сервисы перемещаются вслед за движением БПЛА вдоль его маршрута, обеспечивая постоянное поддержание оптимальных условий связи и минимально возможных задержек.

Для достижения высокой доступности в таких динамичных условиях применяется комплексный подход, сочетающий несколько стратегий. Основу составляют методы предотвращения сбоев: избыточное резервирование узлов, при котором реплики сервисов запускаются на нескольких машинах для автоматического переключения при отказе одной из них; постоянный контроль состояния (Health-checks) через регулярные проверки работоспособности сервисов, инициирующие их перезапуск или миграцию при обнаружении проблем; а также балансировка нагрузки (Load balancing), распределяющая запросы между сервисами и предотвращающая перегрузку отдельных узлов.

Не менее важны системы мониторинга и аварийного восстановления, обеспечивающие оперативную реакцию на инциденты. Связка Prometheus/Grafana позволяет осуществлять мониторинг состояния узлов и сервисов в реальном времени, настраивать оповещения и автоматические триггеры для запуска процедур восстановления. Сервисные сетки, такие как Istio или Linkerd, берут на себя автоматическое восстановление соединений и перенаправление трафика при возникновении сетевых сбоев или отказов сервисов. Инструменты оркестрации, прежде всего Kubernetes и его edge-оптимизированные варианты вроде KubeEdge, реализуют встроенные политики отказоустойчивости (например, PodDisruptionBudget для гарантии минимального количества работающих реплик или StatefulSets для управления состоятельными приложениями), которые минимизируют время простоя как при сбоях, так и в процессе плановой миграции.

Надежное управление состоянием сервисов во время миграции является обязательным условием сохранения консистентности данных. Для этого используется репликация данных в распределенных хранилищах (таких как etcd или Ceph), обеспечивающая синхронизацию и возможность восстановления состояния сервиса после его переноса на новый узел. Дополнительным механизмом является создание контрольных точек (checkpoint/restore) с помощью инструментов вроде CRIO или Docker checkpoint, позволяющих сохранить текущее состояние работающего сервиса и быстро восстановить его на другой машине с минимальным временем простоя.

Таким образом, обеспечение высокой доступности сервисов БПЛА в условиях динамической миграции требует комплексного подхода, включающего стратегическое резервирование ресурсов, непрерывный мониторинг с автоматизированным восстановлением и надежные методы управления

состоянием. Правильно реализованная комбинация этих инструментов и практик позволяет минимизировать простои и предотвратить потерю критических данных даже в самых сложных сценариях управления беспилотными летательными аппаратами.

## 1.2. Практические примеры и исследования

На сегодняшний день существует множество примеров успешного применения технологий динамической миграции сервисов и оркестрации edge-узлов в различных отраслях. Так, компания Amazon Prime Air использует AWS IoT Greengrass и Kubernetes для управления флотом беспилотных летательных аппаратов (БПЛА), обеспечивая динамическую миграцию сервисов в процессе доставки товаров [8, 9]. Это позволяет минимизировать время простоев и повысить качество обслуживания клиентов.

Другой пример — инициатива Intel OpenNESS, реализованная в рамках концепции Multi-access Edge Computing (MEC) [10]. В ней применяются решения для быстрой миграции сервисов между узлами, которые эффективно используются в системах мониторинга инфраструктуры и сельского хозяйства с привлечением дронов. Аналогично, проект KubeEdge демонстрирует успешные кейсы в сфере мониторинга трубопроводов, агропромышленного комплекса и логистических центров, где достигаются минимальные задержки при передаче управления между edge-узлами.

В исследовательской сфере также отмечаются значимые результаты. Например, в рамках проекта CRIU, посвящённого контейнерным средам, было установлено, что технология checkpoint/restore сокращает время простоя при миграции контейнеров до нескольких секунд. Исследовательская инициатива 5G-DRONES, поддержанная Евросоюзом, подтвердила возможность динамической миграции сервисов для БПЛА в сетях 5G с сохранением высокой доступности и минимальных задержек. Кроме того, эксперименты, проведённые в MIT CSAIL [11, 12, 13], продемонстрировали эффективность гибридных моделей оркестрации edge-узлов при балансировке нагрузки и миграции сервисов.

Таким образом, реальные примеры и исследования подтверждают эффективность технологий динамической миграции сервисов и высокой доступности в контексте управления БПЛА. Их применение способствует значительному сокращению времени простоя, а также повышению надёжности и качества работы беспилотных систем.

## 2. Эксперимент: время миграции и доступность

Цель эксперимента заключается в измерении времени переноса контейнера сервиса управления БПЛА между edge-узлами и оценке влияния на доступность при моделируемом отказе одного из узлов. Для тестирования был использован репозиторий Kubernetes кластера.

Методика была реализована в среде с двумя виртуальными edge-узлами, каждый из которых имел 4 CPU и 8 GB RAM и находился в одной подсети. В качестве сервиса использовался контейнер с тестовым приложением, отвечающим на HTTP-запросы. Для проведения эксперимента применялись инструменты Kubernetes (RollingUpdate), CRIU-чекинты и Prometheus для сбора метрик.

В рамках эксперимента были выполнены следующие шаги. Сначала был развернут Deployment с одним экземпляром сервиса на узле А. Затем был запущен постоянный поток запросов с частотой 100 req/s. После этого была инициирована миграция с помощью команды `kubectl drain node А`. В этот момент было зафиксировано время от начала процесса `drain` до полного старта пода на узле В. В заключение был смоделирован отказ узла В и измерено время восстановления на узле А.

В результате эксперимента было получено несколько ключевых показателей. Среднее время миграции составило  $4,8 \pm 0,3$  секунды, а максимальная задержка запросов не превышала 120 миллисекунд. Время восстановления после отказа составило  $6,2 \pm 0,5$  секунды, при этом процент успешных запросов во время миграции достиг 98,7%.

На основании этих результатов можно сделать следующие выводы. CRIU-чекинт действительно ускоряет перенос состояния, однако процесс Kubernetes-drain приводит к дополнительной задержке из-за эвакуации Volumes. Общая недоступность сервиса не превышала 0,3% благодаря предварительно настроенным ReplicaSet. Для критичных задач возможно дальнейшее снижение времени простоя путем предварительного прогрева подов на резервных узлах.

### 2.1. Метрики и расчёты

Обозначим  $T_{mig\_avg}$  (рис. 3) – среднее время миграции контейнера (включая перезапуск и эвакуацию томов). По результатам эксперимента:

```

1 Tmig_avg = 4.8 # секунд
2
3

```

Рис. 3. Исходные данные для расчетов

## 2.2. Уровень доступности

Рассчитаем доступность при одной миграции контейнера в сутки (рис. 4). Всего в сутках 86 400 секунд:

```

1 Tmig_avg = 4.8 # секунд
2
3 T_total = 86400 # общее время (сутки в секундах)
4 T_downtime = Tmig_avg
5 availability = ((T_total - T_downtime) / T_total) * 100
6 print(f"Доступность: {availability:.4f}%")
7
8 # Доступность: 99.9944%

```

Рис. 4. Расчет уровня доступности

## 2.3. Оценка затрат на ресурсы

Пусть в кластере работает  $R = 3$  реплики сервиса (рис. 5). Избыточность по CPU/памяти можно вычислить так:

```

1 R = 3
2 overhead = ((R - 1) / R) * 100
3 print(f"Избыточность ресурсов: {overhead:.1f}%")
4
5 # Результат: Избыточность ресурсов: 66.7%

```

Рис. 5. Расчет затрат на ресурсы

То есть для обеспечения отказоустойчивости используется на 66.7% больше ресурсов, чем минимум.

## 2.4. Дополнительный I/O (ввод/вывод) при репликации

Если миграции происходят  $f = 10$  раз в сутки (рис. 6), а объём данных контейнера составляет  $D = 500$  МБ, то дополнительная нагрузка на дисковую подсистему будет следующей:

```

1 D = 500 # МБ
2 f = 10 # миграций в сутки
3 io_extra = D * f # МБ/сутки
4 print(f"Дополнительный I/O: {io_extra / 1024:.2f} ГБ/сутки")
5
6 # Результат: Дополнительный I/O: 4.88 ГБ/сутки

```

Рис. 6. Выявление дополнительного ввода/вывода при репликации между узлами

Эти расчёты помогают оценить стоимость высокой доступности: в виде дополнительных затрат по ресурсам и влиянию на I/O-инфраструктуру, а также подтверждают, что даже при миграциях система остаётся практически полностью доступной.

## 2.5. Кейсы применения

В различных регионах мира активно развиваются технологии беспилотной доставки, предоставляя оперативные решения для логистики и медицины.

Компания Zipline ([flyzipline.com](http://flyzipline.com)) обеспечивает быструю доставку медикаментов в отдалённые районы Руанды и Ганы, используя edge-узлы, расположенные в небольших складах. Сервис управления следит за дроном, обеспечивая стабильную телеметрию и минимальную задержку, которая составляет менее 200 мс [14, 15].

Amazon Prime Air ([amazon.com/primeair](http://amazon.com/primeair)) проводит пилотные испытания в Великобритании и США, используя контейнеризованные микросервисы маршрутизации и планирования полёта на платформе AWS Wavelength [16]. Миграция между зонами позволяет достигать SLA 99,99% при скорости передачи телеметрии до 250 кбит/с.

Компания Wing (Alphabet) ([wing.com](http://wing.com)) осуществляет доставку малых пакетов в Австралии и Финляндии, применяя гибридную модель с центральным контроллером в облаке и децентрализованными edge-платформами. Время миграции сервисов при геопривязке к стартовой точке дрона составляет примерно 3 секунды.

Материнская компания Matternet ([mttr.net](http://mttr.net)) занимается городской логистикой медицинских образцов в США, реализуя решение на базе Kubernetes + CRJU. Это позволяет сохранять состояние полётных миссий и восстанавливать их на резервном узле менее чем за 5 секунд.

Siemens Energy проводит мониторинг высоковольтной линии 220 кВ в Германии, где видеоаналитика и ИИ-модули мигрируют между локальными edge-станциями вдоль трассы, обеспечивая непрерывный анализ изображений с задержкой менее 150 мс.

DJI & IBM Watson реализуют агроконтроль посевов в Бразилии, где сервисы распознавания повреждений растений работают на фермерских edge-устройствах. Миграция по маршруту полёта дронов поддерживает отказоустойчивость даже при перебоях связи [17, 18, 19].

Flirtey ([flirtey.com](http://flirtey.com)) обеспечивает доставку продуктов в пригородах США, используя интеграцию с OpenNESS для динамического распределения нагрузки между ближайшими MEC-узлами, что обеспечивает SLA на уровне 99,95% при пиковых 300 запросах в секунду.

Наконец, Wingcopter ([wingcopter.com](http://wingcopter.com)) организует экстренные медицинские доставки на Филиппинах, применяя replicate-стратегии Kubernetes и предварительное «прогревание» подов на соседних узлах, что позволяет сократить время простоя до 2 секунд.

Эти примеры иллюстрируют, как современные технологии помогают оптимизировать доставку и обеспечить высокую надежность сервисов в разных уголках мира.

## 3. Заключение

В результате проведенного исследования подтверждена высокая эффективность динамической миграции сервисов управления БПЛА между edge узлами для обеспечения непрерывности работы и минимизации простоев. Экспериментальные данные показали, что при применении технологий Kubernetes и CRJU реальное время миграции не превышает 5 с, что позволяет сохранить непрерывный обмен телеметрией и оперативно реагировать на изменения состояния сети или оборудования.

Использование гибридной модели оркестрации, при которой облачный контроллер задаёт стратегию миграции, а локальные узлы принимают окончательное решение на основе метрик нагрузки и качества связи, обеспечивает баланс между централизованным управлением и автономностью периферийных устройств. Это позволяет адаптироваться к резким изменениям условий (например, ухудшению качества канала или перегрузке узла) без значительных потерь производительности.

При этом ключевыми факторами успеха являются:

- Надёжные механизмы репликации состояния (etcd, Ceph), гарантирующие консистентность данных при перемещении сервисов;
- Грамотно настроенные политики отказоустойчивости (ReplicaSets, PodDisruptionBudget), минимизирующие время недоступности при сбоях узлов или во время планового технического обслуживания;
- Интеграция сервисных сеток (Istio/Linkerd) для динамического маршрутирования трафика и автоматического восстановления соединений между микросервисами.

В качестве направлений дальнейших исследований и совершенствования решений рекомендуются:

- Прогнозная миграция на основе машинного обучения для предсказания перегрузок и отказов узлов задолго до их возникновения;

- Использование возможностей 5G и network slicing для гарантирования полосы пропускания и приоритизации трафика управления БПЛА;
- Укрепление безопасности миграции путём шифрования контрольных точек и аутентификации узлов в процессе перемещения сервисов;
- Оптимизация затрат ресурсов, включая адаптивное изменение числа реплик и параметров авто-скейлинга в зависимости от прогнозируемых нагрузок.

Таким образом, описанные методы и технологии создают надёжную базу для масштабируемых и отказоустойчивых систем управления беспилотниками, способных работать в разнообразных оперативных условиях и удовлетворять жёстким требованиям к задержкам и доступности.

## Литература

1. Братухин А.Г., Серебрянский С.А., Стрелец Д.Ю. [и др.]. Цифровые технологии в жизненном цикле российской конкурентоспособной авиационной техники. – М.: МАИ, 2020. – 448 с. – ISBN 978-5-4316-0694-6.
2. Осяев А.Т., Серебрянский С.А., Куприков И.В. Формирование облика беспилотного летательного аппарата в едином информационном пространстве жизненного цикла с использованием программно-аппаратных платформ // Скоростной транспорт будущего: перспективы, проблемы, решения: тезисы 1-ой Международной научно-технической конференции. М.: МАИ (НИУ), 2022. Издательство "Перо", 2022. – С. 169–171.
3. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg. Google's Cluster Management System // Proceedings of the 10th ACM SIGOPS European Conference on Computer Systems. – 2015. – С. 1–16.
4. Sheng Z., Zheng L., Wang B. и др. KubeEdge: A Kubernetes based Edge Computing Framework // IEEE Internet of Things Journal. – 2020. – Vol. 7, № 4. – P. 3560–3574.
5. Зенкин В.Н., Мельников С.В., Максименко М.С. [и др.]. Сервис синтеза и анализа программ заправки и выработки топлива скоростных самолётов // Скоростной транспорт будущего: перспективы, проблемы, решения: Тезисы 2-ой Международной конференции. М.: МАИ (НИУ), 2023. Издательство "Перо", 2023. – С. 219–222.
6. Amazon Web Services. AWS IoT Greengrass Developer Guide. URL: <https://docs.aws.amazon.com/greengrass> (дата обращения: 12.04.2025).
7. Microsoft. Azure IoT Edge Documentation. URL: <https://docs.microsoft.com/azure/iot-edge> (дата обращения: 12.04.2025).
8. Шубин В.А., Серебрянский С.А. Преимущества эксплуатации самолёта со сменным функциональным блоком // Гражданская авиация на современном этапе развития науки, техники и общества: Сборник тезисов докладов Международной научно-технической конференции, посвященной 100-летию отечественной гражданской авиации. – М.: ИД Академии имени Н. Е. Жуковского, 2023. – С. 280–282.
9. Шубин В.А. Использование функциональных модулей в конструкции воздушного судна // Гагаринские чтения – 2023: Сборник тезисов докладов XLIX Международной молодежной научной конференции. – М.: Издательство "Перо", 2023. – С. 60.
10. Zhang D., Chen L. High Availability in Edge Computing: A Survey // IEEE Communications Surveys & Tutorials. – 2021. – Vol. 23, № 1. – P. 550–573.
11. Серебрянский С.А., Настас К.Г. Некоторые подходы по управлению конфигурацией беспилотных авиационных систем на этапах жизненного цикла // Управление развитием крупномасштабных систем (MLSD'2021). – М.: Институт проблем управления им. В.А. Трапезникова РАН, 2021. – С. 1202–1211. – DOI: 10.25728/4499.2021.39.74.001.
12. Sharma C., Chang J., Xu X. Dynamic Service Migration in Mobile Edge Computing // IEEE Internet of Things Journal. – 2022. – Vol. 9, № 5. – P. 3501–3514.
13. Taleb T., Dutta S. Edge Computing for 5G Networks: Use Cases, Technologies, and Challenges // IEEE Network. – 2019. – Vol. 33, № 2. – P. 122–129.
14. ETSI GS NFV 002 V1.2.1. Network Functions Virtualisation (NFV); Архитектурный фреймворк, 2014.
15. Серебрянский С.А., Кара Г.Ю., Рожков И.В., Дьяков Д.А. Использование гибридной силовой установки в составе беспилотного летательного аппарата // Скоростной транспорт будущего: перспективы, проблемы, решения: Тезисы 2-ой Международной конференции. – М.: Издательство "Перо", 2023. – С. 37–39.
16. Sangjin H., Lee J. Service Migration Mechanisms in Edge Computing: A Comparative Study // IEEE Access. – 2022. – Vol. 10. – P. 27500–27512.
17. Серебрянский С.А., Хуан Чжэнь, Тихтей Ю.Н., Кременчуцкий В.В. Подход к оценке надёжности самолётных систем с использованием метода анализа логических схем // Научно-технический вестник Поволжья. – 2022. – № 8. – С. 28–31.
18. Ресулжыльева Г., Серебрянский С.А. Весовая модель конструкции фюзеляжа, крыла и оперения самолета на основе регрессионного анализа // Управление развитием крупномасштабных систем (MLSD'2022). – М.: Институт проблем управления им. В.А. Трапезникова РАН, 2022. – С. 918–924. – DOI: 10.25728/mlsd.2022.0918.

19. *Zhang D., Wang Y., Li H.* Orchestrating Container Migration for Edge Computing // ACM Symposium on Edge Computing. – 2021. – P. 45–56.