Virtualized Computing and Communication Service Provision in Large-Scale Software-Defined Satellite Networks

Wei Ding[a], Binquan Guo[b,c,d*], Zehui Xiong[d]

[a] Xi'an International Studies University, P. R. China
[b] Xidian University, P. R. China
[c] Tianjin Artificial Intelligence Innovation Center (TAIIC), P. R. China
[d] Singapore University of Technology and Design, Singapore

Author's email: bqguo@stu.xidian.edu.cn
Google Scholar: https://sites.google.com/view/binquanguo/home

August.8 2024

Outline

Introduction

Motivation

- As reported by ITU, $\approx$ 2.9 billion people ($\approx$ 37% of the world's population) still do not have access to Internet, and $>$ 70% of the surface of the earth has no terrestrial network (e.g., in ocean, dessert).
- One direction: Companies like SpaceX are developing mega-satellite constellations with computing and communication resources to provide direct-to-ground services for various mobile applications.

Software Defined Satellite Networks v.s. Traditional Ones (Iridium)

- Enhancements: inter-satellite links, on-board computing (Tianzhi 1, Tiansuan), and large scale (Starlink plans 42,000 satellites).
- Advantages: low-latency, resource virtualization (More flexible).
- Challenges: High dynamics, diverse communication and computing services, delay sensitive and fast response time. + Key issue: Service provision methods for real time applications in dynamic SN are absent.

Introduction

Software Defined SN:

1. **Network resources:** $w_{\mathbb{S}_6}^{f_1} = 3$ means node $\mathbb{S}_6$ can serve 3 computing services of type-f_1, and $q_{(\mathbb{U}_1,\mathbb{S}_3)}^{h_1} = 2$ means link $(\mathbb{U}_1, \mathbb{S}_3)$ can serve 2 communication services of type-h_1.
2. **App requirements:** Each application $\mathcal{A} = (\mathbb{U}_s, \mathbb{U}_d, f_a, h_a, D_a)$ has a pair of source and sink UE, the specified services $\{f_a, h_a\}$ and delay requirement D_a.

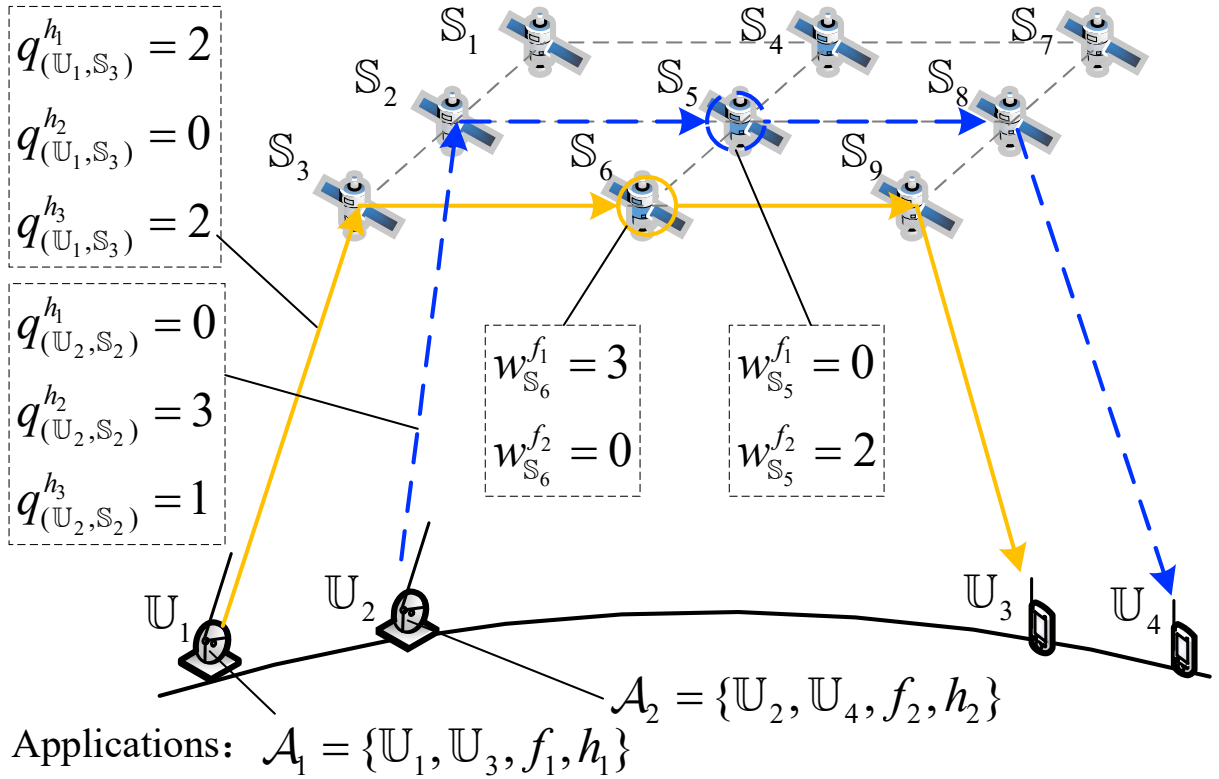

Figure: The scenario of service provision for real-time applications in software defined SNs.

Related work

Category 1: Network function virtualization in the SNs

1 L. Bertaux, et. al, "Software defined networking and virtualization for broadband satellite networks," IEEE Commun. Mag., 2015.

2 G. Wang, et. al, "SFC-based service provisioning for reconfigurable space-air-ground integrated networks," IEEE JSAC, 2020.

3 Z. Jia, et. al, "VNF-based service provision in software defined leo satellite networks," IEEE TWC, 2021.

Category 2: Routing and resource allocation for in the SNs

4 S. Zhou, et. al, "Bidirectional mission offloading for agile space-air-ground integrated networks," IEEE Wireless Commun., 2019.

5 G. Araniti, et. al, "Contact graph routing in DTN space networks: overview, enhancements and performance," IEEE Commun. Mag., vol. 53, no. 3, pp. 38–46, Mar. 2015.

6 Y. Hu, et. al, "Time-deterministic networking for satellite-based internet-of-things services: Architecture, key technologies, and future directions," IEEE Netw., pp. 1–7, Mar. 2024.

System Model and Problem Formulation

1. **Network scenario:** We consider a software-defined SN composed of satellites and user equipments (UEs), denoted as $\mathcal{S} = \{\mathbb{S}_1, ..., \mathbb{S}_P\}$ and $\mathcal{U} = \{\mathbb{U}_1, ..., \mathbb{U}_K\}$, respectively. Each satellite is equipped with computing and communication hardwares. Computing and communication resources on each satellite can be configured as virtual services to support various tasks by taking advantage of SDN, NFV and micro-services techniques. ISLs are supported.
2. **Service model:** Denote the set of computing services as $\mathcal{F} = \{f_1, f_2, ., f_N\}$, where the maximum number of applications that can request service f_l on satellite $\mathbb{S}i$ is $w_{\mathbb{S}_i}^{f_l} \in \{0, 1, 2, 3, ...\}$. Similarly, the set of communication services is represented as $\mathcal{H} = \{h_1, h_2, ., h_L\}$, where the upper limit for a communication service $g_r \in \mathcal{H}$ in a link $(\mathbb{O}_i, \mathbb{O}_j)$ is set as $q_{(\mathbb{O}_i, \mathbb{O}_j)}^{h_r} \in \{0, 1, 2, 3, ...\}$.
3. **APP model:** $\mathcal{A} = (\mathbb{U}_s, \mathbb{U}_d, f_a, h_a, D_a)$ denotes an application from $\mathbb{U}_s$ to $\mathbb{U}_d$, requesting computing service f_a and communication service h_a, with the end-to-end delay not exceeding D_a.

System Model and Problem Formulation

Time-varying graph-based representation model

1. **Time division method**: The time horizon $\mathcal{T} = [0, T]$ is cut into variable length time windows. Each time window $\tau = [t_s, t_e]$.
2. **Snapshot graph**: The SN in time window τ can be viewed as a snapshot graph $\mathcal{G}^\tau = (\mathcal{V}^\tau, \mathcal{L}^\tau)$. $\mathcal{V}^\tau$ denotes the node set, and $\mathcal{L}^\tau = \{(\mathbb{O}_i, \mathbb{O}_j)\}$ is the set of communication links within τ.
3. **Resource attributes**:
   Each computing service specifies distinct resource requirements, such as {1 core CPU, 2 GB Memory}, and each communication service specifies its required bandwidth (unit: Mbps). The maximum computational service capacity of a satellite $\mathbb{S}_i$ is represented as an N-dimensional set $\mathcal{W}_{\mathbb{S}_i} = \{w_{\mathbb{S}_i}^{f_1}, w_{\mathbb{S}_i}^{f_2}, ..., w_{\mathbb{S}_i}^{f_N}\}$. The maximum allowable call numbers of communication services of a link $(\mathbb{O}_i, \mathbb{O}_j)$ are collected into a L-dimensional set $\mathcal{Q}_{(\mathbb{O}_i,\mathbb{O}_j)} = \{q_{(\mathbb{O}_i,\mathbb{O}_j)}^{h_1}, q_{(\mathbb{O}_i,\mathbb{O}_j)}^{h_2}, ., q_{(\mathbb{O}_i,\mathbb{O}_j)}^{h_L}\}$, where $|\mathcal{Q}_{(\mathbb{O}_i,\mathbb{O}_j)}| = L$. The propagation delay of link $(\mathbb{O}_i, \mathbb{O}_j)$ can be calculated and denoted as $D_{(\mathbb{O}_i,\mathbb{O}_j)}^\tau$ (unit: ms).

System Model and Problem Formulation

Decision variables

- $x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} = 1$: indicates that the service path allocated will traverse through $(\mathbb{O}_i, \mathbb{O}_j)$.

Parameters

- $\mathcal{A} = (\mathbb{U}_s, \mathbb{U}_d, f_a, h_a, D_a)$: denotes an application from $\mathbb{U}_s$ to $\mathbb{U}_d$.
- f_a: the required computing service,
- h_a: the required communication service,
- D_a: the maximum acceptable end-to-end delay from $\mathbb{O}_s$ to $\mathbb{O}_d$.
- $D^{\tau}_{(\mathbb{O}_i,\mathbb{O}_j)}$: The link delay of link $(\mathbb{O}_i, \mathbb{O}_j)$ in time window τ.
- $q^{h_l}_{(\mathbb{O}_i,\mathbb{O}_j)}$: the capacity of communication service h_l in satellite $\mathbb{S}_i$.
- $w^{f_x}_{\mathbb{S}_i}$: the capacity of computing service f_x in satellite $\mathbb{S}_i$.

Objective function

- The objective is to minimize the end-to-end delay, i.e.,

$$\min \sum_{(\mathbb{O}_i,\mathbb{O}_j)\in\mathcal{L}^{\tau}} x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \cdot D^{\tau}_{(\mathbb{O}_i,\mathbb{O}_j)}$$

A. Basic Constraints

- 1. Source node constraints:

$$\sum_{\mathbb{O}_k \in \mathcal{V}^\tau - \{\mathbb{U}_s\}} x^\tau_{\mathbb{U}_s,\mathbb{O}_k} = 1 - \sum_{\mathbb{O}_k \in \mathcal{V}^\tau - \{\mathbb{U}_s\}} x^\tau_{\mathbb{O}_k,\mathbb{U}_s} = 1. \tag{1}$$

- 2. Sink node constraints:

$$\sum_{\mathbb{O}_k \in \mathcal{V}^\tau - \{\mathbb{U}_d\}} x^\tau_{\mathbb{O}_k,\mathbb{U}_d} = 1 - \sum_{\mathbb{O}_k \in \mathcal{V}^\tau - \{\mathbb{U}_d\}} x^\tau_{\mathbb{U}_d,\mathbb{O}_k} = 1. \tag{2}$$

- 3. Constraints for relay satellites:

$$\sum_{\mathbb{O}_k:(\mathbb{O}_k,\mathbb{O}_\xi)\in\mathcal{L}} x^\tau_{\mathbb{O}_k,\mathbb{O}_\xi} \leq 2, \forall \mathbb{O}_\xi \in \mathcal{V}^\tau - \{\mathbb{U}_s, \mathbb{U}_d\}. \tag{3}$$

$$\sum_{\mathbb{O}_k:(\mathbb{O}_\xi,\mathbb{O}_k)\in\mathcal{L}} x^\tau_{\mathbb{O}_\xi,\mathbb{O}_k} \leq 2, \forall \mathbb{O}_\xi \in \mathcal{V}^\tau - \{\mathbb{U}_s, \mathbb{U}_d\}. \tag{4}$$

$$\sum_{\mathbb{O}_k:(\mathbb{O}_k,\mathbb{O}_\xi)\in\mathcal{L}^\tau} x^\tau_{\mathbb{O}_k,\mathbb{O}_\xi} = \sum_{\mathbb{O}_k:(\mathbb{O}_\xi,\mathbb{O}_k)\in\mathcal{L}^\tau} x^\tau_{\mathbb{O}_\xi,\mathbb{O}_k}. \tag{5}$$

B. Service provision constraints

- 4. Communication service constraint:

$$x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \leq q^{h_a}_{(\mathbb{O}_i,\mathbb{O}_j)}, \forall(\mathbb{O}_i,\mathbb{O}_j) \in \mathcal{L}^{\tau}. \tag{6}$$

- 5. Computing service constraint:

$$\sum_{\mathbb{O}_\xi \in \mathcal{S}} [(\sum_{\mathbb{O}_k:(\mathbb{O}_k,\mathbb{O}_\xi)\in\mathcal{L}^{\tau}} x^{\tau}_{\mathbb{O}_k,\mathbb{O}_\xi}) \cdot w^{f_a}_{\mathbb{O}_\xi}] \geq 1. \tag{7}$$

- 6. Service delay constraint:

$$\sum_{(\mathbb{O}_i,\mathbb{O}_j)\in\mathcal{L}^{\tau}} x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \cdot D^{\tau}_{(\mathbb{O}_i,\mathbb{O}_j)} \leq D_a. \tag{8}$$

B. Constraints Against Sub-tours in the Service Path Caused by Computing Service Provision

- 7. We introduce variable $y^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \in \{0,1,2,3,...\}$ for each link $(\mathbb{O}_i,\mathbb{O}_j)$. $y^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} = z > 0$ indicates $(\mathbb{O}_i,\mathbb{O}_j)$ is the z-th hop in the service path.
- 8.Link the variables x and y:

$$x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \leq y^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \leq x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \cdot M, \tag{9}$$

$$y^{\tau}_{\mathbb{O}_i,\mathbb{O}_j} \leq \sum_{(\mathbb{O}_i,\mathbb{O}_j)\in\mathcal{L}_{\tau}} x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j}, \tag{10}$$

Here, $M = 10^6$ is a large constant for logical constraint reformulation.

- 9.Source and destination UE constraints:

$$\sum_{\mathbb{O}_k\in\mathcal{V}^{\tau}-\{\mathbb{U}_s\}} y^{\tau}_{\mathbb{U}_s,\mathbb{O}_k} = 1, \tag{11}$$

$$\sum_{\mathbb{O}_k\in\mathcal{V}^{\tau}-\{\mathbb{U}_d\}} y^{\tau}_{\mathbb{O}_k,\mathbb{U}_d} = \sum_{(\mathbb{O}_i,\mathbb{O}_j)\in\mathcal{L}_{\tau}} x^{\tau}_{\mathbb{O}_i,\mathbb{O}_j}. \tag{12}$$

C. Sub-tour removal constraints

- Service path continuity constraints: We use a sub-tour removal method to ensure path service continuity by maintaining the sequential hop rule for communication links of every chosen relay satellites.

$$\sum_{\mathbb{O}_k:(\mathbb{O}_k,\mathbb{O}_\xi)\in\mathcal{L}^\tau}(y^\tau_{\mathbb{O}_k,\mathbb{O}_\xi}+x^\tau_{\mathbb{O}_k,\mathbb{O}_\xi})=\sum_{\mathbb{O}_k:(\mathbb{O}_\xi,\mathbb{O}_k)\in\mathcal{L}^\tau}y^\tau_{\mathbb{O}_\xi,\mathbb{O}_k}. \tag{13}$$

- Hop number order correctness constraint: $\forall(\mathbb{O}_i,\mathbb{O}_j)\in\mathcal{L}^\tau$,

$$y^\tau_{\mathbb{O}_i,\mathbb{O}_j}+(1-x^\tau_{\mathbb{O}_i,\mathbb{O}_j})M\geq(\sum_{\mathbb{O}_k:(\mathbb{O}_k,\mathbb{O}_i)\in\mathcal{L}^\tau}y^\tau_{\mathbb{O}_k,\mathbb{O}_\xi})-y^\tau_{\mathbb{O}_j,\mathbb{O}_i}+1. \tag{14}$$

Constraint (14) means when an outgoing link of an intermediate node $\mathbb{O}_i$, say $(\mathbb{O}_i,\mathbb{O}_j)$, is included in the selected path with hop number $y^\tau_{\mathbb{O}_i,\mathbb{O}_j}$, then $\mathbb{O}_i$ must have an incoming link with hop number equals $y^\tau_{\mathbb{O}_i,\mathbb{O}_j}-1$, and this incoming link is not the reverse link of $(\mathbb{O}_j,\mathbb{O}_i)$.

System Model and Problem Formulation

Virtual Service Provision Problem Formulation

With the objective of reducing the overall service delay, the virtual service provision problem is formulated as:

$$\mathbf{P1}: \min \sum_{(\mathbb{O}_i,\mathbb{O}_j)\in\mathcal{L}^\tau} x^\tau_{\mathbb{O}_i,\mathbb{O}_j} \cdot D^\tau_{(\mathbb{O}_i,\mathbb{O}_j)}$$

$$\text{s.t. } (1)-(14).$$

Analysis and Observation

- **P1** is an integer linear programming (ILP) problem.
- **P1** can be solved by commercial integer programming solvers, such as Gurobi, using the branch and bound (B&B) method.
- The B&B for solving **P1** has a worst-case time complexity of $\mathcal{O}(2|\mathcal{L}^\tau| \cdot B^{2|\mathcal{L}^\tau|})$, which grows exponentially with the number of links $|\mathcal{L}^\tau|$. It is necessary to exploit the special structure of **P1** and obtain more efficient methods.

Proposed Virtual Service Provision Scheme

Optional-1: KSP-based virtual service provision algorithm

1. **Input:** The snapshot graph $\mathcal{G}^{\tau}$ and an application $\mathcal{A}$.
2. **Output:** The virtual service provision scheme p_*.
3. Initialize $\mathcal{G}^{\tau}_{h_a} \leftarrow \mathcal{G}^{\tau}$, $k = 1$, and $D_{p_k} = -\infty$.
4. **for** $(\mathbb{O}_i, \mathbb{O}_j) \in \mathcal{L}^{\tau}$ **do**
5. Remove link $(\mathbb{O}_i, \mathbb{O}_j)$ from $\mathcal{G}^{\tau}_{h_a}$ **if** $q^{h_a}_{(\mathbb{O}_i, \mathbb{O}_j)} < 1$.
6. **while** $D_{p_k} \leq D_a$ **do**
7. Determine the k-th path p_k and its delay D_{p_k} on $\mathcal{G}^{\tau}_{h_a}$.
8. **for** each computing satellite $\mathbb{S}_{\xi} \in p_k$ **do**
9. **if** $w^{f_a}_{\mathbb{S}_{\xi}} \geq 1$ **then**
10. $p_* \leftarrow p_k$, **break**
11. $k = k + 1$
12. **return** p_*.

Proposed Virtual Service Provision Scheme

The drawbacks of the KSP-based algorithm

- **Sub-optimality:** Only allow simple paths. Applications may be rejected though there is non-simple path with repeated nodes.
- **Instability:** The running time is related to both the value of D_a and the number of satellites, which is unstable and pseudo-polynomial.
- **Non Scalability:** For large scale SNs with 1000+ nodes, the number of paths is extremely large, making it time consuming.

Complexity analysis

- The classical KSP algorithm requires about $O(K|\mathcal{V}^\tau|^3)$ computational steps to find K paths. However, its practicality diminishes in scenarios involving a complete graph of size $|\mathcal{V}^\tau|$ and an unlimited delay threshold, where **Algorithm 1** exhibits worst-case complexity at $O(|\mathcal{V}^\tau|! \cdot |\mathcal{V}^\tau|^3)$. This factorial growth in complexity makes **Algorithm 1** impractical for real-world deployment.

Proposed Virtual Service Provision Scheme

Option-2: The service-aware optimal provision algorithm

1. **Input:** The snapshot graph $\mathcal{G}^{\tau}$ and an application $\mathcal{A}$.
2. Initialize empty node set $\mathcal{V}_{f_a}^{\tau}$, empty graph $\mathcal{G}_{h_a}^{\tau}$, and service delay $D_{p_*} = +\infty$.
3. **for** $(\mathbb{O}_i, \mathbb{O}_j) \in \mathcal{L}^{\tau}$ **do**
4. **if** $q_{(\mathbb{O}_i,\mathbb{O}_j)}^{h_a} \geq 1$ **then**
5. Add $(\mathbb{O}_i, \mathbb{O}_j)$ to $\mathcal{G}_{h_a}^{\tau}$, $\mathbb{O}_i$ to $\mathcal{V}_{f_a}^{\tau}$ if $w_{\mathbb{O}_i}^{f_a} \geq 1$, and $\mathbb{O}_j$ to $\mathcal{V}_{f_a}^{\tau}$ if $w_{\mathbb{O}_j}^{f_a} \geq 1$.
6. **if** $\mathcal{V}_{f_a}$ is empty **then**, **return** -1.
7. Run the shortest path algorithm twice on $\mathcal{G}_{h_a}^{\tau}$ and $\mathcal{G}_{h_a}^{\tau}$, respectively.
8. **for** each candidate computing satellite $\mathbb{S}_i \in \mathcal{V}_{f_a}^{\tau}$ **do**
9. $p_{(\mathbb{U}_s \to \mathbb{S}_i \to \mathbb{U}_d)} = p_{(\mathbb{U}_s, \mathbb{S}_i)} + \bar{p}_{(\mathbb{U}_d, \mathbb{S}_i)}$, $D_{p_{(\mathbb{U}_s \to \mathbb{S}_i \to \mathbb{U}_d)}} = D_{p_{(\mathbb{U}_s, \mathbb{S}_i)}} + D_{p_{(\mathbb{U}_d, \mathbb{S}_i)}}$.
10. **if** $D_{p_{(\mathbb{U}_s \to \mathbb{S}_i \to \mathbb{U}_d)}} < D_{p_*}$ **then** update $p_* = p_{(\mathbb{U}_s \to \mathbb{S}_i \to \mathbb{U}_d)}$, $D_{p_*} = D_{p_{(\mathbb{U}_s \to \mathbb{S}_i \to \mathbb{U}_d)}}$.
11. **return** The service-aware optimal provision scheme p^*.

Proposed Virtual Service Provision Scheme

The advantages of the proposed SASP algorithm

- **Polynomial:** Node and link resources filtering + running Dijkstra's algorithm twice + path concatenating and selection.
- **Optimal:** Both simple service paths and non-simple service paths in feasible solution space are covered.

Complexity analysis

- Using the SASP method involves filtering all available links and nodes, which requires a total of $O(|\mathcal{L}^\tau|)$ iterations. The time complexity of Dijkstra's algorithm for computing shortest paths is $\mathcal{O}(|\mathcal{L}^\tau| + |\mathcal{V}^\tau| \log |\mathcal{V}^\tau|)$. Hence, **Algorithm 2** demonstrates polynomial complexity at $\mathcal{O}(3|\mathcal{L}^\tau| + (2 \log |\mathcal{V}^\tau| + 1)|\mathcal{V}^\tau|)$ in worst case, showcasing its scalability with increasing network size.

Simulation

A. Scenarios

- A software-defined Starlink constellation with 2,000 LEO satellites, currently the largest LEO satellite system.

B. Parameters

- UE locations: Kashi $(39.5°N, 76°E)$, Sanya $(18°N, 109.5°E)$, Beijing $(40°N, 116°E)$ and Xi'an $(34.27°N, 108.93°E)$.
- The link delays of ISLs and USLs: $[5, 15]$ ms.
- Both of computing and communication services have 10 types, with 10% of satellites support for computing.
- We test 5,000 randomly generated applications. Each application has two UEs requesting specific services from sets $\mathcal{F}$ and $\mathcal{H}$, with end-to-end delays randomly ranging from 20 to 150 ms.

C. Algorithms

- ILP-based, KSP-based, our proposed SASP.
- All the three algorithms are implemented using Python.

Simulation Results

1. Figure 2 shows average running times versus satellite numbers. KSP and SASP greatly outperform the ILP, with SASP faster than KSP.
2. Figure 3 shows more satellites decrease average delays. SASP achieves lower delays than KSP when satellite numbers is lower ($= 100$).

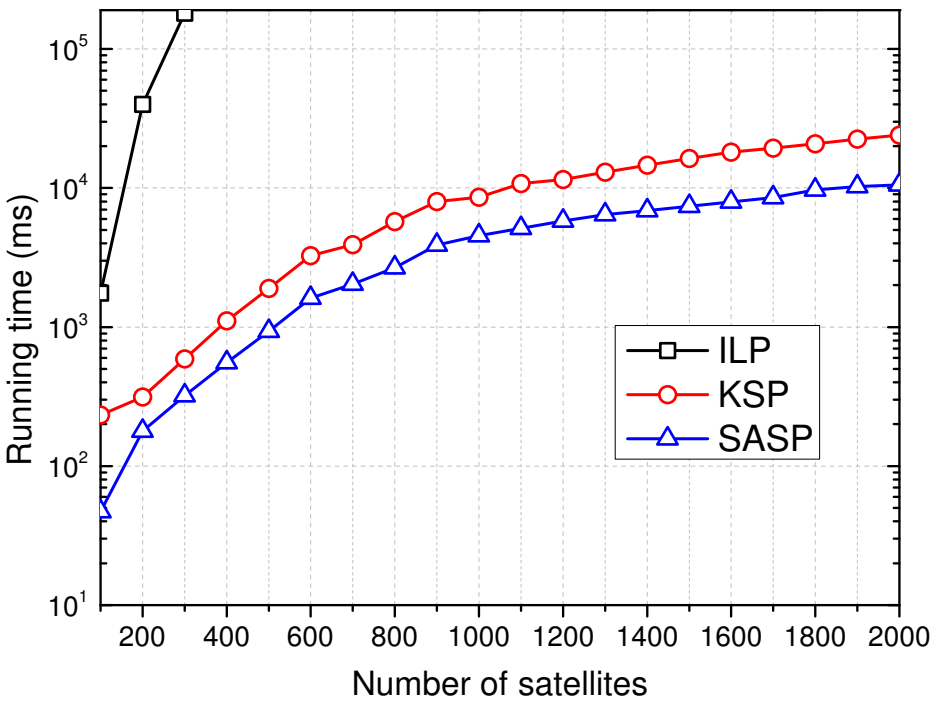

Figure: Running times versus different network sizes.

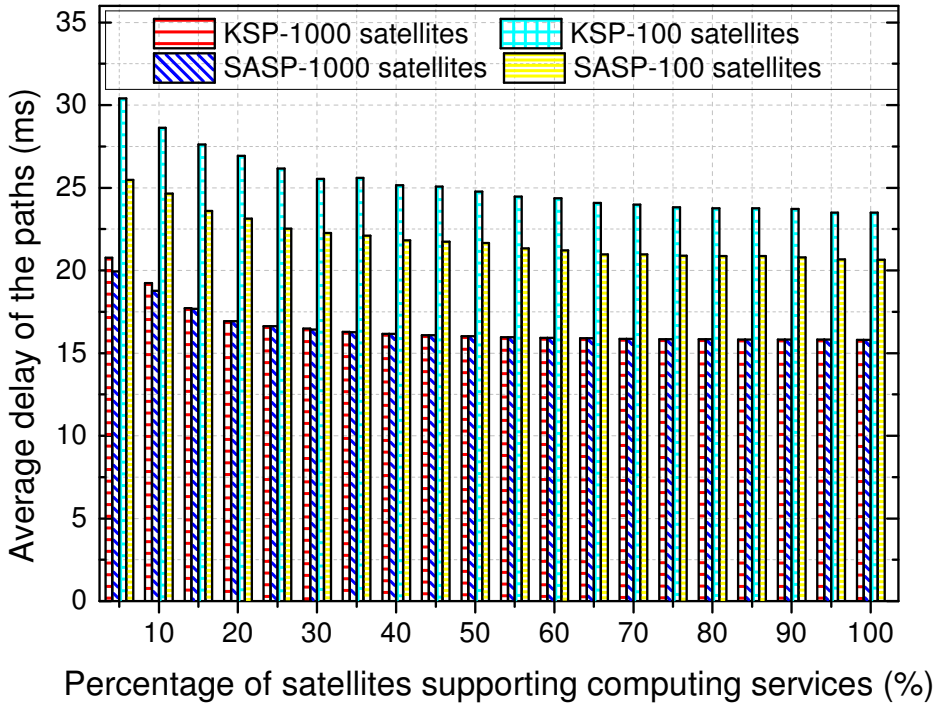

Figure: Path delays versus percentages of computing satellites.

Conclusion

1. We formulate the virtualized computing and communication service provision problem in SN as an ILP, which incurs $\mathcal{O}(2|\mathcal{L}^\tau| \cdot B^{2|\mathcal{L}^\tau|})$ time complexity and is intractable in practice.
2. By adopting a KSP-based algorithm, the time complexity becomes $O(|\mathcal{V}^\tau|! \cdot |\mathcal{V}^\tau|^3)$, which is sub-optimal and has drawbacks, resulting in compromised performance in practice.
3. To overcome this, we further design a graph-based algorithm by exploiting the special structure of the solution space, which can obtain the optimal solution in polynomial time with a computational complexity of $\mathcal{O}(3|\mathcal{L}^\tau| + (2\log|\mathcal{V}^\tau| + 1)|\mathcal{V}^\tau|)$.
4. Simulations conducted on starlink constellation with thousands of satellites demonstrate that the SASP is better than KSP and ILP.
5. Our scheme can support the deployment of emerging applications (e.g., the large AI models, in-orbit semantic communication, AI-based video/image processing, etc) as virtual services on satellites in future.

Thank you for listening!

Feel free to contact us via Wechat ↑ or E-mail ↓ if you have any questions.

Email:*bqguo@stu.xidian.edu.cn*

Google page: https://sites.google.com/view/binquanguo/home