Application of Evolutionary Mechanism to Dynamic Virtual Network Function Placement

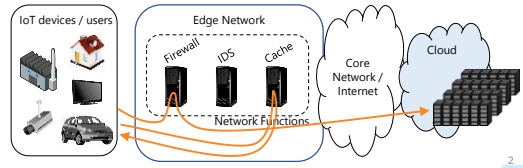
Mari Otokura[1], Kenji Leibnitz[2], Yuki Koizumi[1], Daichi Kominami[1], Tetsuya Shimokawa[2], Masayuki Murata[1]

[1] Osaka University, Japan

[2] NICT, Japan

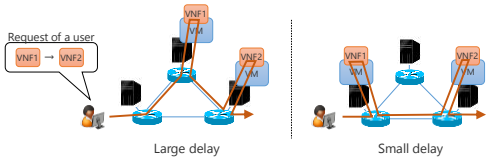
Research Background

- Communication services are becoming more diverse and dynamic
  - Internet of Things (IoT) is currently being realized
- Communication service providers will offer flexible and dynamic *network functions*
  - Network functions: process packets in the "middle" of networks
    - Examples: firewalls, intrusion detection systems (IDS), caches
  - Network functions have been implemented in hardware
    - Not flexible and not dynamic



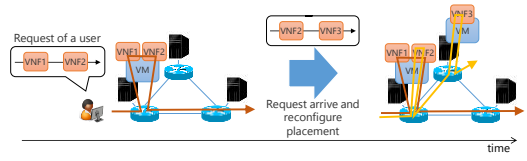
Network Function Virtualization (NFV)

- NFV implements network functions in software
- Virtual Network Functions (VNFs):
  - Network functions virtualized by NFV
  - Run on Virtual Machines (VMs)
- VNF placement problem
  - Decision of placement of VNFs on Physical Machines (PMs) in physical networks



Dynamic VNF Placement Problem

- Reconfiguration of VM/VNF placements on physical network when requests for VNF chains arrive/depart
- Main requirement for placement computation: short calculation time
- Solving optimization problem at every request change:
  - Difficult to realize since even the static VNF placement problem is NP-hard



Our Objective and Approach

- Objective
  - Propose method which meets the requirements of the dynamic VNF placement problem
- Approach
  - Utilization of knowledge from biological evolution under varying environments
    - When organisms evolve in varying environments:
      - They become robust to environmental changes [6]
      - Their evolution speeds up [2]

[15] N. Kashtan and U. Alon, "Spontaneous Evolution of Modularity and Network Motifs," *PNAS*, vol. 102, no. 39, pp. 13773–13778, Sep. 2005.

[16] N. Kashtan, E. Noor, and U. Alon, "Varying Environments Can Speed Up Evolution," *PNAS*, vol. 104, no. 34, pp. 13711–13716, Aug. 2007.

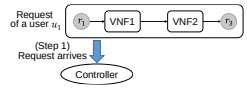
System Model

(Step 1) *Request* for a *VNF chain* from a user arrives

(Step 2) Controller splits VNFs of chain into *components*
- Components: small VNF software modules [4]

(Step 3) Controller places components on PMs and assigns cores to them

(Step 4) Controller decides routes of traffic for chains through required components



[13] ETSI, GS NFV-SWA 001 - V1.1.1 - Network Functions Virtualisation (NFV); Virtual Network Functions Architecture, Dec. 2014.

Formulation of VNF Placement Problem

- Minimize average delay and number of cores under constraints (2)-(4):
  - (2) Each VNF must have sufficient number of cores to guarantee requested transmission speed for each user
  - (3) VMs cannot use more cores than available on PMs
  - (4) Components on VM cannot use more cores than possibly available on VM

$$\text{minimize} \quad \hat{d} + W \cdot \sum_{i,k} m_{i,k} \qquad (1)$$

subject to:

$$T_a \cdot \frac{v_a}{s} \leq n_{k,j,a} \cdot C \quad \forall k,j,a \qquad (2)$$

$$\sum_k m_{i,k} \leq N_i \quad \forall i \qquad (3)$$

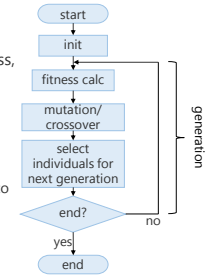
$$\sum_{j,a} n_{k,j,a} \leq m_{i,k} \quad \forall k,i \qquad (4)$$

variables $m_{i,k}, n_{k,j,a}, p^u_{(r,r')}$

$m_{i,k}$: number of cores which the VM k occupies on PM i
$n_{k,j,a}$: number of cores which component j of VNF i occupies on VM k

Evolutionary Algorithm (EA)

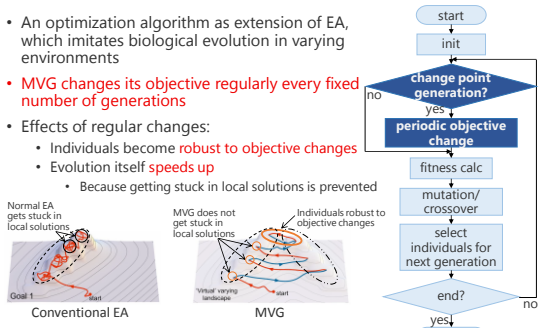
- An optimization heuristic which imitates biological evolution
- Every loop (generation), EA calculates fitness, applies mutations and crossovers to individuals, selects better individuals, and uses these as population in the next generation
  - Individual: representation of solutions
    - Population: set of all individuals
  - Fitness: goodness of individuals according to objectives
  - Mutation: randomly and slightly change individual
  - Crossover: randomly combine two individuals



[15] N. Kashtan and U. Alon, "Spontaneous Evolution of Modularity and Network Motifs," *PNAS*, vol. 102, no. 39, pp. 13773–13778, Sep. 2005.

Modularly Varying Goals (MVG) [15]

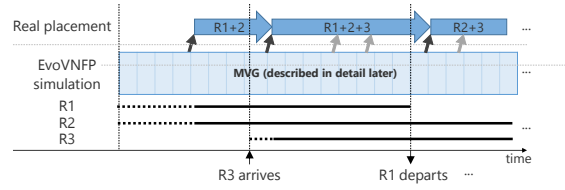
- An optimization algorithm as extension of EA, which imitates biological evolution in varying environments
- MVG changes its objective regularly every fixed number of generations
- Effects of regular changes:
  - Individuals become robust to objective changes
  - Evolution itself speeds up
    - Because getting stuck in local solutions is prevented



[16] N. Kashtan, E. Noor, and U. Alon, "Varying Environments Can Speed Up Evolution," *PNAS*, vol. 104, no. 34, pp. 13711–13716, Aug. 2007.

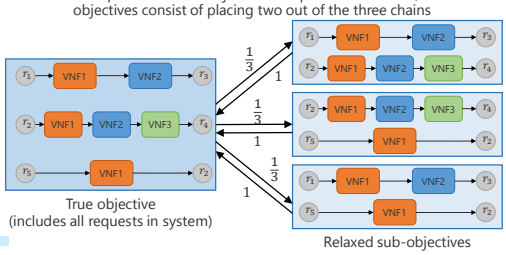
Evolvable VNF Placement (EvoVNFP)

- Dynamic VNF placement method addressing dynamic arrivals/departures of requests of VNF chains
  - Calculation of placements by MVG (described in detail later)
  - When simulations generate placements which meets predetermined objectives, the controller implements generated placements as real ones



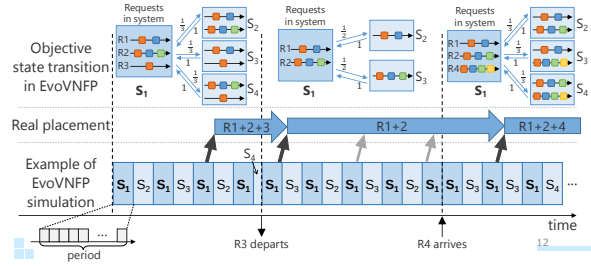
Objective Changes of MVG in EvoVNFP

- Objective changed every fixed number of generations (= period) according to remaining requests
  - Switching between true objective and relaxed sub-objectives
    - Generates adaptive structure and speeds up evolution
  - Example: When true objective is to place three chains, the relaxed objectives consist of placing two out of the three chains



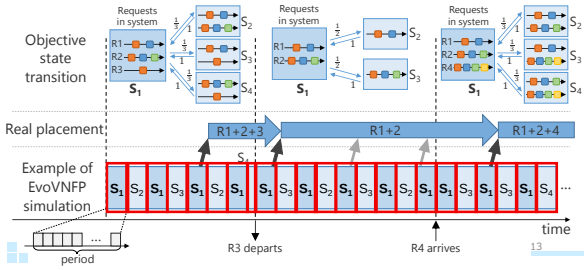
Example of EvoVNFP Simulation

- Intentionally use EA without re-initialization of population when objectives change
- Intentionally change objectives every period



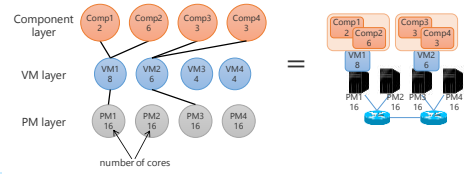
Example of EvoVNFP Simulation

- Intentionally use EA without re-initialization of population when objectives change
- Intentionally change objectives every period



Individuals and Mutations

- Example structure of an individual (see figure below):
  - Individual represents placement in network
    - Connection between VM layer and component layer: allocation of component on VM
    - Connection between PM layer and VM layer: allocation of VM on PM
- Mutation: randomly change one element of an individual
  - Change connections or the number of cores saved in nodes



Fitness Function

- Function to calculate how well placements adapt to objectives
  - If individuals can be converted to placements:
    - Smaller average delay of chains and number of used cores → better fitness
  - Otherwise:
    - Fitness is a negative value
    - Smaller number of elements in individuals violating the constraints → better fitness

$$F = \begin{cases} \left(\frac{\hat{d}}{d_{max}} + \frac{W(\sum_{l,k} m_{l,k})}{c_{max}}\right)^{-1} & \text{if individuals can be converted to placements} \\ -Z & \text{otherwise} \end{cases}$$

d_{max}: reference value of delay
c_{max}: maximum number of cores
Z: number of elements which violate the constraints

Simulation Settings

- Physical network: 5 routers, 10 PMs, each PM has 16 cores
- Requests: tuples consisting of ingress router, egress router, VNF chain, and transmission rate
  - Example: $(r_1, r_3, \{VNF1 \rightarrow VNF2\}, 200\ \text{Mbps})$
- Reference methods for comparison:
  - Conventional EA (Conv): normal EA at every arrival/departure of requests
  - Random Immigrant GA (RandImm) [19]
    - RandImm initializes randomly selected individuals after mutation step instead of changing objectives every period as in EvoVNFP
- Parameters:
  - Population size: 1000, elite size: 100, mutation probability: 0.8
  - Period (EvoVNFP): 20 generations, replacement rate (RandImm): 0.3
  - Fixed load of system: $Load = arrival\ rate \times sojourn\ time = 0.2$

[19] J. Grefenstette, "Genetic Algorithms for Changing Environments," in *Proceedings of PPSN 1992*. Elsevier, Sep. 1992, pp. 137–144.

Evaluation Metrics

- Failure probability
  - Probability of finding no feasible solution until the next objective change
- Number of generations
  - Number of generations until obtaining the first feasible solution
- Cost of reconfigurations
  - Weighted average of the number of reconfigurations [20, 21]

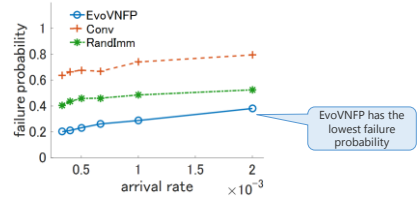
Migrations of VMs/components in a router	30
Migrations of VMs/components between routers	120
Resizing of VMs/components, removal of VMs	1
Addition of VMs	60

- Performance of generated placements
  - Sum of the number of cores assigned to VMs (system performance)
  - Average delay of chains (user performance)

[20] J. Barrera, M. Ruiz, and L. Velasco, "Orchestrating Virtual Machine Migrations in Telecom Clouds," in *Proceedings of OFC 2015*, Mar. 2015, pp. 1–3.
[21] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A Cost-Aware Elasticity Provisioning System for the Cloud," in *Proceedings of ICDCS 2011*, Jun. 2011, pp. 559–570.

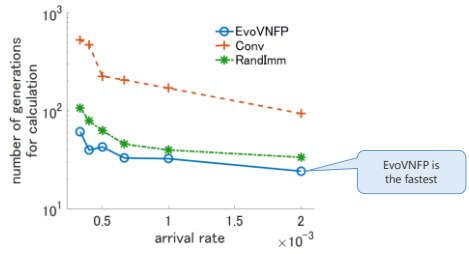
Failure Probability

- EvoVNFP shows best performance among all three methods
  - This means that EvoVNFP can follow the dynamics of request arrivals/departures



Number of Generations

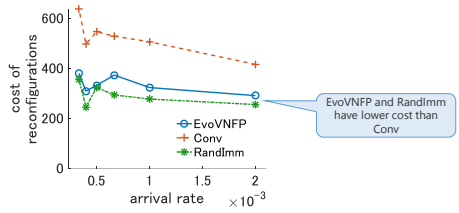
- EvoVNFP can generate solutions with the fewest generations
  - This means that EvoVNFP can generate solution in a short time



19

Cost of Reconfigurations

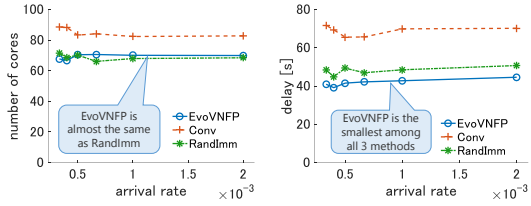
- Methods without re-initialization of population when requests arrive/depart have lower cost
  - EvoVNFP and RandImm have similar cost, lower than Conv



20

Performance of Generated Placements

- EvoVNFP can generate better solutions than comparative methods
  - This means that EvoVNFP can generate high-performance solutions while reducing the failure probability



21

Summary and Future Work

- Summary
  - Proposal of dynamic VNF placement method named EvoVNFP
    - EvoVNFP generates placements which meet user requests by MVG
    - When requests arrive/depart, EvoVNFP runs EA without reinitializing population
    - EvoVNFP switches between real objectives and relaxed sub-objectives every fixed number of generations
  - Confirmation of effectiveness of EvoVNFP by computer simulations
    - EvoVNFP reduces failure probability of not generating valid placements and also reduces time to generate solutions
    - Furthermore, EvoVNFP generates high-performance placements
- Future work
  - Application of different evaluation metrics

22