Dynamic Resource Provisioning and Scheduling in SDN/NFV-Enabled Core Networks

by

Kaige Qu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Amiya Nayak Professor, School of Electrical Engineering & Computer Science University of Ottawa
Supervisor:	Weihua Zhuang Professor, Department of Electrical and Computer Engineering University of Waterloo
Internal Member:	Oleg Michailovich Associate Professor Department of Electrical and Computer Engineering University of Waterloo
Internal Member:	Xiaodong Lin Adjunct Associate Professor Department of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Jun Liu Associate Professor, Department of Applied Mathematics University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The service-oriented fifth-generation (5G) core networks are featured by customized network services with differentiated quality-of-service (QoS) requirements, which can be provisioned through network slicing enabled by the software defined networking (SDN) and network function virtualization (NFV) paradigms. Multiple network services are embedded in a common physical infrastructure, generating service-customized network slices. Each network slice supports a composite service via virtual network function (VNF) chaining, with dedicated packet processing functionality at each VNF. For a network slice with a target traffic load, the end-to-end (E2E) service delivery is enabled by VNF placement at NFV nodes (e.g., data centers and commodity servers) and traffic routing among corresponding NFV nodes, with static resource allocations. To provide continuous QoS performance guarantee over time, it is essential to develop dynamic resource management schemes for the embedded services experiencing traffic dynamics in different time granularities during virtual network operation. In this thesis, we focus on processing resources and investigate three research problems on dynamic processing resource provisioning and scheduling for embedded delay-sensitive services, in presence of both large-timescale traffic statistical changes and bursty traffic dynamics in smaller time granularities.

In problem I, we investigate a dynamic flow migration problem for multiple embedded services, to accommodate the large-timescale changes in the average traffic rates with average E2E delay guarantee, while addressing a trade-off between load balancing and flow migration overhead. We develop optimization problem formulations and efficient heuristic algorithms, based on a simplified M/M/1 queueing model with Poisson traffic arrivals. Motivated by the limitations of Poisson traffic model, in problem II, we restrict to a local network scenario and study a dynamic VNF scaling problem based on a real-world traffic trace with nonstationary traffic statistics in large timescale. Under the assumption that the nonstationary traffic trace can be partitioned into non-overlapping stationary traffic segments with unknown change points in time, a change point detection driven traffic parameter learning and resource demand prediction scheme is proposed, based on which dynamic VNF migration decisions are made at variable-length decision epochs via deep reinforcement learning. The long-term trade-off between load balancing and migration overhead is studied. A fractional Brownian motion (fBm) traffic model is employed for each detected stationary traffic segment, based on properties of Gaussianity and self-similarity of the real-world traffic. In Problem III, we focus on a sufficiently long time duration with given

VNF placement and stationary traffic statistics, and study a delay-aware VNF scheduling problem to coordinate VNF scheduling for multiple services, which achieves network utility maximization with timely throughput guarantee for each service, in presence of bursty and unpredictable small-timescale traffic dynamics, while using a realistic *state-of-the-art* time quantum (slot) for CPU processing resource scheduling among VNF software processes. Based on the Lyapunov optimization technique, an online distributed VNF scheduling algorithm is derived, which greedily schedules a VNF at each NFV node based on a weight incorporating the backpressure-based weighted differential backlogs with the downstream VNF, the service throughput performance indicated by virtual queue lengths, and the packet delay.

With the proposed dynamic resource management framework, resources can be efficiently and fairly allocated to the embedded services, to avoid congestion and QoS degradation in the presence of traffic dynamics. This research provides some insights in dynamic resource management for delay-sensitive services in a virtualized network environment with CPU processing resources.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Weihua Zhuang, for her professional guidance, constant support, and invaluable suggestions throughout the research work and preparation of this thesis. Her rigorous academic attitude and insightful thoughts always inspire me to do in-depth thinking on my research. Her warmth encouragements help me to face challenges, overcome frustration and keep going with determination. It was my great honor to study and work under her supervision and guidance.

My sincere appreciation also goes to Professor Xuemin (Sherman) Shen for his great support and invaluable suggestions. Many thanks for the wonderful talks by Professor Shen on emerging research topics, research attitudes and presentation skills, which greatly help me to enrich my knowledge and vision.

Specially, I would like to thank Professor Jun Liu, Professor Oleg Michailovich, Professor Xiaodong Lin, and the external examiner, Professor Amiya Nayak, for devoting their invaluable time and serving on my PhD thesis examining committee. Their suggestions, comments and valuable questions have greatly helped me to improve the quality of my thesis.

I would like to thank my colleagues from the BBCR-SDN research sub-group: Dr. Qiang Ye, Dr. Omar Alhussein, Si Yan, Dr. Phu Thinh Do, Dr. Junling Li, Jiayin Chen, Dr. Weisen Shi, Dr. Peng Yang, Dr. Shan Zhang, for their inspiring suggestions, warmth encouragement, and precious friendship. I wish to extend my appreciation to all the other members in the BBCR group with whom I have enjoyed every moment.

Finally, I wish to thank my dearest mother, father, and brother, for their love, support and encouragement. A special thank to my dear husband Wei for his ever-growing love, support, and patience. It is them who make me determined to move forward and never fear hardships on my way.

Dedication

To my beloved parents, Chunfang Yao and Xiangxue Qu,
and my dearest husband, Wei Zhang.

Table of Contents

List of Figures

List of Tables

List of Acronyms

5G	fifth generation
BOCPD	Bayesian online change point detection
CFS	completely fair scheduler
CPU	central processing unit
DQN	deep Q-network
EMBB	enhanced mobile broadband
FBm	Fractional Brownian motion
GPR	Gaussian process regression
GPS	generalized processor sharing
HFM	hybrid flow migration
HTTP	Hypertext Transfer Protocol
IDS	intrusion detection system
InP	infrastructure provider
LBFM	load balancing flow migration
LRD	Long-range dependence
MANO	management and orchestration
MDP	Markov decision process
MIQCP	mixed integer quadratically constrained programming
MMTC	massive machine-type communication
MMPP	Markov-modulated Poisson process
MOFM	minimum overhead flow migration
NAT	network address translator
NFVI	NFV infrastructure
NFVO	NFV orchestrator
NSO	network service orchestrator

OS	operating system
QQ	quantile-quantile
RLS	real time scheduler
RO	resource orchestrator
SFC	service function chain
SLA	service level agreement
TE	traffic engineering
URLLC	ultra reliable low latency communication
VIM	Virtualized infrastructure manager
VNF	virtual network function
VNFM	VNF manager
VR	virtual reality

List of Symbols

$\mathbb{E}$	Expectation over the randomness in packet arrivals, VNF scheduling, and packet processing
$\mathbb{L}(\boldsymbol{\theta})$	Loss function
$\mathbb{M}$	Big-$\mathbb{M}$ constant
$\mathbb{P}$	Probability or likelihood
$\mathbb{1}\{\cdot\}$	The indicator function which is equal to 1 only if the condition inside the bracket is true
$\mathcal{A}$	The set of service access nodes
$\mathcal{B}$	The set of bottleneck NFV nodes for each SFC with largest per-hop delay along the E2E path
$\hat{\mathcal{C}}(k)$	The estimated k-th change point in hour
$\mathcal{C}_M(k)$	The k-th change point in the number of medium time intervals
$\mathcal{C}_S(k)$	The k-th change point in the number of small time intervals
$\mathcal{E}$	The set of virtual links
$\mathcal{H}_r$	The set containing the index of VNFs of service $r \in \mathcal{R}$, i.e., $\{1, \cdots, H_r\}$
$\mathcal{H}_1^{(r)}(\tau)$	Subset of VNFs in service $r \in \mathcal{R}$ including all the unscheduled VNFs during time slot τ
$\mathcal{H}_2^{(r)}(\tau)$	Subset of VNFs in service $r \in \mathcal{R}$ including all the scheduled VNFs with packet rushing opportunity during time slot τ
$\mathcal{H}_3^{(r)}(\tau)$	Subset of VNFs in service $r \in \mathcal{R}$ including all the scheduled VNFs without packet rushing opportunity during time slot τ
$\mathcal{M}_h^{(r)}$	Set of packet residual lifetime at VNF $V_h^{(r)}$ without packet rushing
$\tilde{\mathcal{M}}_h^{(r)}$	Set of packet residual lifetime at VNF $V_h^{(r)}$ with packet rushing
$\mathcal{N}$	The set of NFV nodes in a core network
$\mathcal{N}_C$	A set of candidate NFV nodes in a local network segment

$\mathcal{N}_O$	The set of overloaded NFV nodes
$\mathcal{N}_U$	The set of underloaded NFV nodes
$\mathcal{N}_{U,O}$	The set of underloaded NFV nodes with SFC category II
$\mathcal{N}_{U,U}$	The set of underloaded NFV nodes without SFC category II
$\mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}_k)$	$\mathcal{Q}$ function
$\mathcal{R}$	The set of services
$\mathcal{V}_n$	A set containing the index of all VNFs placed at NFV node $n \in \mathcal{N}$, with $(r, h) \in \mathcal{V}_n$ denoting VNF $V_h^{(r)}$
Γ	Number of time slots for VNF scheduling
$\Delta\left(\boldsymbol{\Theta}(\tau)\right)$	A one-step conditional Lyapunov drift for time slot τ
$\boldsymbol{\Theta}(\tau)$	The combined queue vector at time slot τ, i.e., $\boldsymbol{\Theta}(\tau) = [\boldsymbol{q}(\tau), \boldsymbol{W}(\tau), \boldsymbol{H}(\tau)]$
Λ	The long-term average traffic rate in packet/s for a certain VNF
Ξ	An auxiliary continuous decision variable (MIQCP)
Υ_l	Threshold for the absolute difference between the most probable run lengths at two medium time intervals
Υ_d	Threshold for the normalized absolute difference between the estimated mean plus standard deviation corresponding to the most probable run lengths
$\Phi_1^{(r)}(\tau)$	A function for service $r \in \mathcal{R}$ in the upper bound of the conditional Lyapunov drift-plus-penalty during time slot τ
$\Phi_2^{(r)}(\tau)$	A function for service $r \in \mathcal{R}$ in the upper bound of the conditional Lyapunov drift-plus-penalty during time slot τ
Ψ	Covariance matrix
Ω_r	The maximal tolerable service downtime for SFC $r \in \mathcal{R}$
$\alpha_{\mathcal{Q}}$	Learning rate in deep-Q learning
$\tilde{\alpha}_n(k)$	A positive vertical delay scaling factor less than 1 which is applied to hop delay bounds of multiple SFCs belonging to category III on NFV node $n \in \mathcal{N}_{U,U}$ during interval k

$\tilde{\beta}^{(r)}(k)$	A horizontal delay scaling factor larger than 1 which is applied to hop delay bounds of SFC r in category III on NFV nodes in $\mathcal{N}_{U,O}$ during interval k
γ	Discount factor
δ_j	TD-error of transition j
ϵ_1	A constant with $0 < \epsilon_1 \ll 1$
ϵ_2	Exploring probability in deep Q-learning
$\epsilon_{2,\Delta}$	A step size for decreasing exploration probability ϵ_2
ε	The maximum delay violation probability (or packet dropping ratio)
ε_r	The maximum packet dropping ratio of service $r \in \mathcal{R}$
$\boldsymbol{\zeta}(k)$	Auxiliary nonnegative continuous decision variable set (MIQCP)
$\eta(k)$	The maximum NFV node loading factor during interval k
$\eta_n(k)$	The resource loading factor of NFV node n during interval (epoch) k
$\eta_n^B(k)$	Background resource loading factor of NFV node $n \in \mathcal{N}_C$ at epoch k
$\eta_n^{rh}(k)$	A ratio between resources occupied by VNF $V_h^{(r)}$ and resource capacity of NFV node n during interval k
η_{th}	The threshold for NFV node loading factor
η_Δ	Step size for updating η_{th}
η_Δ^ϵ	A constant denoting the required precision of $\eta(k)$ and η_{th}
η_U	An upper limit for the maximum NFV node loading ratio without penalty
$\boldsymbol{\theta}$	Weights of evaluation DQN
$\hat{\boldsymbol{\theta}}$	Weights of target DQN
ϑ	A utility importance parameter that balances the importance between utility maximization and queue backlog reduction
$\boldsymbol{\iota}(k)$	Auxiliary nonnegative continuous decision variable set (MIQCP)
$\lambda(k)$	FBm traffic parameter for the k-th stationary traffic segment: mean packet arrivals in each small time interval
$\lambda^{(r)}(k)$	The average Poisson packet arrival rate in packet/s for SFC $r \in \mathcal{R}$ during interval k
μ	Mean of packet arrivals in a medium time interval
ν	Standard deviation of packet arrivals in a medium time interval

$\boldsymbol{\xi}(k)$	Auxiliary binary decision variable set (MIQCP)
$\pi(\boldsymbol{s})$	Policy
$\boldsymbol{\varpi}(k)$	Auxiliary nonnegative continuous decision variable set (MIQCP)
ρ	The resource overprovisioning ratio used in the simulations for VNF scheduling
$\boldsymbol{\varrho}(k)$	Auxiliary nonnegative continuous decision variable set (MIQCP)
$\sigma(k)$	FBm traffic parameter for the k-th stationary traffic segment: standard deviation of packet arrivals in each small time interval
$\boldsymbol{\varsigma}(k)$	Auxiliary nonnegative continuous decision variable set (MIQCP)
τ	Index of the τ-th VNF scheduling time slot
$\upsilon_{h,max}^{(r)}$	The maximum packet processing rate (in packet/s) for VNF $V_h^{(r)}$
$\upsilon_h^{(r)}(\mathtt{t})$	The actual packet processing rate (in packet/s) for VNF $V_h^{(r)}$ at time $\mathtt{t} \in [0, T]$ for time slot τ, where $\mathtt{t}$ is reset to 0 at the beginning of time slot τ
$\boldsymbol{\upsilon}_h^{(r)}$	A vector of actual packet processing rates for VNF $V_h^{(r)}$
$\phi(\cdot)$	A strictly increasing and concave utility function of throughput
φ	A weight for the virtual queue length $W^{(r)}(\tau)$ in the Lyapunov function $L\left(\boldsymbol{\Theta}(\tau)\right)$, i.e., $\varphi = \left[\max_{r\in\mathcal{R}} a_r\right]\left[\max_{r\in\mathcal{R}, h\in\mathcal{H}_r} P_h^{(r)}\right]$
$\bar{\chi}_r$	The infinite-horizon time average expectations of $\chi^{(r)}(\tau)$ for service $r \in \mathcal{R}$, i.e., $\bar{\chi}_r = \lim_{\Gamma\to\infty} \frac{1}{\Gamma} \sum_{\tau=0}^{\Gamma-1} \mathbb{E}\left\{\chi^{(r)}(\tau)\right\}$
$\chi^{(r)}(\tau)$	An auxiliary decision variable for service $r \in \mathcal{R}$ at time slot τ in the VNF scheduling algorithm, with $\chi^{(r)}(\tau) \in [0, A_{max}^{(r)}]$
ψ	Covariance function
ω_1, ω_2, ω_3	Three weighting factors to control the priority of the three objectives in Problem I, with $\omega_1 + \omega_2 + \omega_3 = 1$
$\omega^{(\mathtt{C})}$	A weighting factor in cost c_k
$\omega^{(\mathtt{TD})}$	A parameter controlling the relative importance of TD error and penalty in priority p_j
$\omega_{h,\mathbb{U}}^{(r)}(\tau)$	The scheduling weight for one urgent packet at VNF $V_h^{(r)}$ during time slot τ

$\omega_{h,\mathbb{N}}^{(r)}(\tau)$	The scheduling weight for one non-urgent packet at VNF $V_h^{(r)}$ during time slot τ
$\omega_h^{(r)}(\tau)$	The VNF scheduling weight for VNF $V_h^{(r)}$ during time slot τ without packet rushing
$\tilde{\omega}_h^{(r)}(\tau)$	The VNF scheduling weight for VNF $V_h^{(r)}$ during time slot τ with packet rushing
$A^{(r)}(\tau)$	The number of packets that arrive at the τ-th time slot for service $r \in \mathcal{R}$
$\mathsf{A}(t)$	The cumulative packet arrivals before small time interval t
$B_{n,n'}^{rh}(k)$	The transmission resource overhead for transferring the state of VNF $V_h^{(r)}$ from NFV node n to NFV node n' during time interval k
B_r	A constant related to the parameters of service $r \in \mathcal{R}$ in the upper bound of the conditional Lyapunov drift-plus-penalty
C_n	The CPU processing capacity of NFV node $n \in \mathcal{N}$ in cycle/s
C_n	The CPU processing resource budget at NFV node $n \in \mathcal{N}$ in cycle per time slot
D	Delay bound in the QoS requirement
D_r	The average end-to-end delay requirement for SFC $r \in \mathcal{R}$
$\mathsf{D}^{rh}(k)$	The per-hop delay requirement for VNF $V_h^{(r)}$ during interval k
$D_h^{(r)}(\tau)$	The number of packets dropped from the queue of VNF $V_h^{(r)}$ during time slot τ
E_k	The prediction error of t_0 traffic samples in the k-th stationary traffic segment
$F^{(r)}(\tau)$	Virtual queue length for service $r \in \mathcal{R}$ at the beginning of time slot τ
G	A graph representing a network of NFV nodes, edge switches, and virtual links $G = \{\mathcal{N} \cup \mathcal{A}, \mathcal{E}\}$
$H(k)$	Hurst parameter for the k-th stationary traffic segment
H_r	The number of VNFs in service $r \in \mathcal{R}$
$I_n^{e,1}$	A binary parameter to indicate whether $n \in \mathcal{N} \cup \mathcal{A}$ is the starting point of virtual link $e \in \mathcal{E}$
$I_n^{e,2}$	A binary parameter to indicate whether $n \in \mathcal{N} \cup \mathcal{A}$ is the ending point of virtual link $e \in \mathcal{E}$

J	Batch size
K	Number of learning steps in an episode
$K_{\boldsymbol{\theta}}$	Number of learning steps to replace $\hat{\boldsymbol{\theta}}$ by $\boldsymbol{\theta}$
$L\left(\boldsymbol{\Theta}(\tau)\right)$	The Lyapunov function which represents a scalar metric of congestion level in the queueing system at the beginning of time slot τ
$\mathtt{M}$	Size of replay memory
M_r	The E2E deadline (in number of time slots) for each packet of service $r \in \mathcal{R}$
$N_m(k)$	The total number of VNF migrations during interval k
$N_e(k)$	The total number of extra virtual links for flow rerouting during interval k
$O(k)$	The objective function for interval k
$P_h^{(r)}$	The processing density of VNF $V_h^{(r)}$ in cycle/packet
P_n^{rh}	The processing density of VNF $V_h^{(r)}$ at NFV node $n \in \mathcal{N}$ in cycle/bit
P_r	The aggregate processing density (in cycle/packet) of service $r \in \mathcal{R}$, i.e., $P_r = \sum_{h \in \mathcal{H}_r} P_h^{(r)}$
$\mathbf{P}_\tau$	An instantaneous VNF scheduling problem for time slot τ
$\mathbf{P}_\infty$	The stochastic VNF scheduling problem with decision variables for Γ ($\Gamma \to \infty$) time slots
$Q_{h,\mathbb{U}}^{(r)}(\tau)$	The total numbers of urgent packets in the queue of VNF $V_h^{(r)}$ during time slot τ
$Q_{h,m}^{(r)}(\tau)$	The length of conceptual queue $Q_{h,m}^{(r)}$ (or the number of packets with a residual lifetime $m \in \mathcal{M}_h^{(r)}$ at VNF $V_h^{(r)}$) at the beginning of time slot τ
$R_h^{(r)}(\tau)$	The number of rushing packets processed by VNF $V_h^{(r)}$ during time slot τ
$R_{h,m}^{(r)}(\tau)$	The number of rushing packets with residual lifetime $m \in \tilde{\mathcal{M}}_h^{(r)}$ processed by VNF $V_h^{(r)}$ during time slot τ
R	Processing resource demand in packet per time unit (small time interval)
$\mathsf{R}(k)$	Processing resource demand in packet/s of the k-th detected stationary traffic segment

R_n	The maximum supporting processing rate (in packet/s) of NFV node n
$S_{h,\mathbb{N}}^{(r)}(\tau)$	The number of non-urgent packets processed at VNF $V_h^{(r)}$ of service $r \in \mathcal{R}$ during time slot τ
$S_{h,\mathbb{U}}^{(r)}(\tau)$	The number of urgent packets processed at VNF $V_h^{(r)}$ of service $r \in \mathcal{R}$ during time slot τ
$S_h^{(r)}(\tau)$	The number of packets processed from the queue of VNF $V_h^{(r)}$ during time slot τ
$S_{h,m}^{(r)}(\tau)$	The number of packets with residual lifetime $m \in \mathcal{M}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ
$\hat{S}_{h,m}^{(r)}(\tau)$	The number of packets with residual lifetime $m \in \mathcal{M}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ if VNF $V_h^{(r)}$ is scheduled
$\tilde{S}_h^{(r)}(\tau)$	The actual number of packets processed by VNF $V_h^{(r)}$ during time slot τ with the consideration of packet rushing
$\tilde{S}_{h,m}^{(r)}(\tau)$	The actual number of packets with residual lifetime $m \in \tilde{\mathcal{M}}_h^{(r)}$ processed by VNF $V_h^{(r)}$ during time slot τ with the consideration of packet rushing
$S_n^{rh}(k)$	The processing rate (in packet/s) allocated to VNF $V_h^{(r)}$ by NFV node n during interval k in Problem I
T	Length of a VNF scheduling time slot in second
T_M	Length of a medium time interval in second
T_S	Length of a small time interval in second
T_n	The CPU polling period in NFV node $n \in \mathcal{N}$
$U_h^{(r)}(k)$	The state size (in bit) of VNF $V_h^{(r)}$ in time interval k
$V_h^{(r)}$	The h-th ($h \in \mathcal{H}_r$) VNF in service $r \in \mathcal{R}$
$\mathtt{W}$	The switching time overhead in a CPU polling period
$W^{(r)}(\tau)$	Virtual queue length for service $r \in \mathcal{R}$ at the beginning of time slot τ
$X_n^{(r)}(k)$	Binary decision variable indicating whether SFC r traverses NFV node n during interval k
$Y_h^{(r)}$	The h-th subflow in SFC $r \in \mathcal{R}$
$Z(t)$	A general fractional Brownian motion (fBm)

a_k	Action at decision epoch k, which is an integer denoting the VNF location during decision epoch k, with $\mathrm{a}_k = n$ if the VNF is placed at NFV node n
$\bar{a}_r$	The mean rate (in packet per time slot) of traffic for service $r \in \mathcal{R}$
b_r	The average packet size in bit for SFC $r \in \mathcal{R}$
c	An arbitrary constant value selected from interval (1,2]
c_k	Cost at learning step k
$c^{(\mathsf{P})}$	A constant representing the level of penalty in the cost
d	A random variable denoting the VNF packet processing delay
$d_n^{rh}(k)$	The (dummy) processing delay on VNF $V_h^{(r)}$ at NFV node n during time interval k
$\bar{d}_r$	The average E2E delay (in second) of the timely delivered packets of service $r \in \mathcal{R}$ to the egress edge switch
e	Virtual link $e \in \mathcal{E}$
$\bar{f}_r$	The throughput (in packet per time slot) of deadline-constrained service $r \in \mathcal{R}$
$\mathtt{f}_1^{(r)}$	A binary flag indicating whether SFC r traverses NFV nodes in $\mathcal{N}_O$
$\mathtt{f}_2^{(r)}$	A binary flag indicating whether SFC r traverses NFV nodes in $\mathcal{N}_{U,O}$
$\mathtt{f}_k^{(\mathsf{P})}$	A binary flag indicating whether there is penalty due to resource overloading at learning step k
$g_{n,n'}^{rh}(k)$	A binary decision variable for whether the mapped NFV node for VNF $V_h^{(r)}$ changes from NFV node n to NFV node n' during time interval k
h	Index of the h-th VNF in a service
i	An integer denoting the i-th medium time interval
k	An integer denoting the k-th decision epoch (or interval, or learning step) corresponding to the k-th stationary traffic segment
l_i	A random variable denoting run length at the i-th medium time interval
m	Packet residual lifetime in the number of time slots
n	NFV node n
$\mathtt{n}_1^{(r)}$	The source node (edge switch) of SFC $r \in \mathcal{R}$
$\mathtt{n}_2^{(r)}$	The destination node (edge switch) of SFC $r \in \mathcal{R}$

o_1	Prioritization level in calculating sampling probability of transition j
o_2	Level of compensation in importance sampling
p_j	Priority of transition j
$q_h^{(r)}(\tau)$	The number of packets in the packet processing queue associated with VNF $V_h^{(r)}$ at the beginning of time slot τ
r	Service $r \in \mathcal{R}$
r_k	Reward at learning step k
$\boldsymbol{s}_k$	State at learning step k
t	An integer denoting the t-th small time interval
$\mathtt{t}$	Time in second
$\mathtt{t}_1$	The transition time instant for the actual packet processing rate of a VNF with packet rushing opportunity
$\mathbf{t}_h^{(r)}$	A vector of time boundaries between the actual packet processing rates for VNF $V_h^{(r)}$
$u_{n,n'}^{rh}(k)$	The (dummy) delay to transfer the state of VNF $V_h^{(r)}$ from NFV node n to NFV node n' during interval k
$\mathtt{v}$	A binary variable indicating whether migrations are required
$\mathtt{w}_j$	Weight for transition j in importance sampling
$\mathrm{w}_n(k)$	A binary decision variable for whether switching happens at NFV node n during interval k
$x_M(i)$	Number of packets in the i-th medium time interval
$x_S(t)$	Number of packets in the t-th small time interval
$\mathrm{x}_n^{rh}(k)$	A binary decision variable for whether VNF $V_h^{(r)}$ is mapped to NFV node n during interval k
x_n^{rh}	A binary VNF placement parameter, with $\mathrm{x}_n^{rh} = 1$ if VNF $V_h^{(r)}$ is placed at NFV node $n \in \mathcal{N}$, and $x_n^{rh} = 0$ otherwise
$y_{nn'}^{rh}(k)$	A binary decision variable for whether subflow $Y_h^{(r)}$ is mapped to an extra virtual link between $n \in \mathcal{N} \cup \mathcal{A}$ and $n' \in \mathcal{N} \cup \mathcal{A}$ during interval k
$\mathtt{y}_k$	Target value at learning step k
$z_h^{(r)}(\tau)$	A binary VNF scheduling decision variable, with $z_h^{(r)}(\tau) = 1$ if VNF $V_h^{(r)}$ is scheduled for packet processing at the corresponding NFV node during time slot τ and $z_h^{(r)}(\tau) = 0$ otherwise

Chapter 1

Introduction

1.1 SDN/NFV-Enabled Core Networks

The service-oriented fifth generation (5G) networks will support new use cases and diverse services with multi-dimensional performance requirements [1]. There are three typical 5G use case families: enhanced mobile broadband (eMBB), massive machine-type communication (mMTC), and ultra reliable low latency communication (uRLLC), whose disparate performance requirements are difficult to be satisfied by the legacy *one-size-fits-all* network architecture. Instead, network slicing is required on a per-service basis, to provide service-level performance guarantees. Multiple network slices with diverse performance requirements are embedded over a common physical infrastructure [2–7]. This requires a flexible and programmable network architecture, with abstraction on both the plane and layer dimensions [8]. Software-defined networking (SDN) brings the plane-dimension abstraction by decoupling the data and control planes. With a global network view and flow awareness brought by SDN, end-to-end (E2E) data delivery paths can be dynamically established by configuring forwarding rules in SDN-enabled switches via southbound protocols such as Openflow, and resources are explicitly allocated to different paths by an SDN controller [9]. Network function virtualization (NFV) provides the layer-dimension abstraction, by abstracting physical resources to virtual resources with a virtualization layer and realizing service-level functionalities [8, 10]. Traditionally, service providers rely on dedicated hardware middleboxes to realize network functions as in-path packet processing units required by a service, such as intrusion detection system (IDS), network address

translator (NAT), firewall, 5G evolved packet core functions, cache, wireless access network optimizer, transcoder, etc. The dedicated hardware middleboxes are expensive and lack flexibility for deployment and management. NFV separates network functions from dedicated hardware to software instances, referred to as virtual network functions (VNFs), hosted in NFV nodes such as commodity servers and data centers. NFV enables cost-effective VNF placement and elastic VNF capacity scaling. Several frameworks have been proposed for SDN-NFV integration, to fully exploit their advantages and provide an integrated architecture with abstractions in both the plane and layer dimensions for customized service provisioning [8, 11]. An extended NFV management and orchestration (MANO) architecture with SDN integration is illustrated in Fig. 1.1 and described as follows.

1.1.1 NFV Infrastructure Domain

Physical Network

A physical network consists of SDN switches and NFV nodes interconnected by physical links. Switches forward traffic from incoming physical links to outgoing physical links. Some switches act as edge switches for service access. NFV nodes, such as commodity servers and data centers, have both forwarding and processing capabilities. The physical network contains a physical resource pool, including transmission resources on physical links and processing resources at NFV nodes. A path in the physical network, i.e., a physical path, is composed of a series of physical links and SDN switches between two NFV nodes or between one NFV node and one edge switch.

Virtual Resource Pool

A logical abstraction of all physical paths with pre-allocated transmission resources between two different NFV nodes or between one NFV node and one edge switch is referred to as a virtual link. The maximum transmission rate supported by a virtual link is the aggregate maximum transmission rate over all its underlying physical paths. Transmission resources on virtual links are seen as virtual resources, since the mapping between virtual links and physical paths is transparent to service flows traversing the virtual links. With network function virtualization, the processing resources at NFV nodes are virtualized and distributed among several VNFs through a virtualization hypervisor. Hence, a virtual

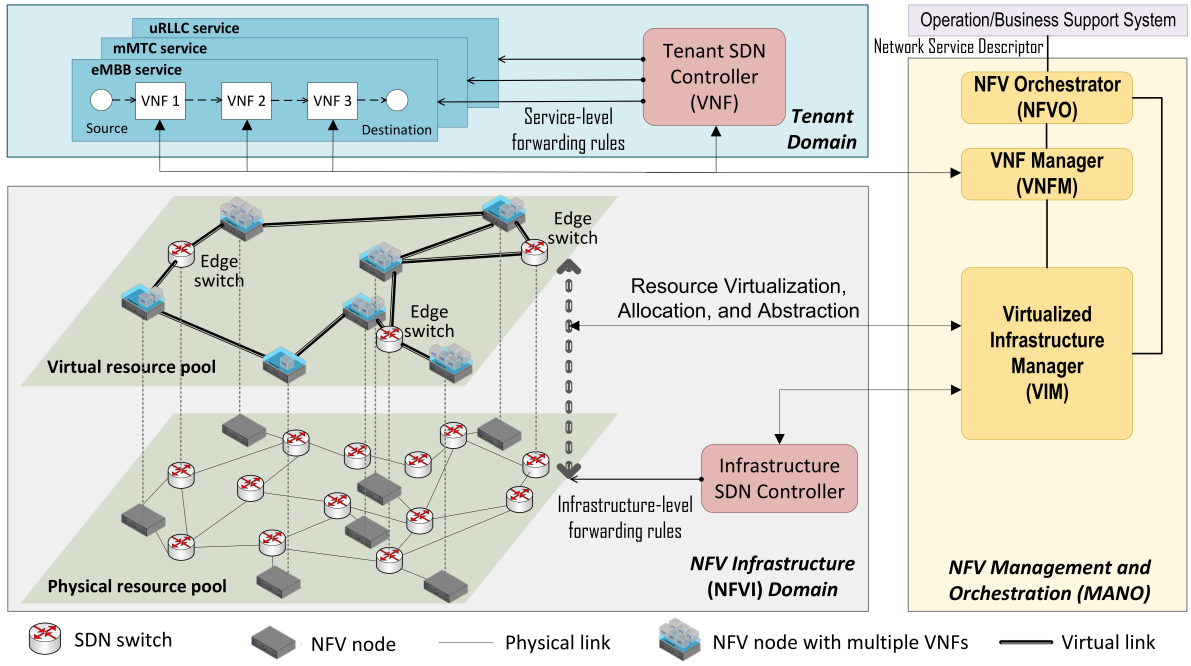


Figure 1.1: An extended NFV MANO architecture with SDN integration.

resource pool containing virtual transmission and processing resources can be abstracted from the physical resource pool, which makes both SDN switches and physical links fully transparent to service flows. A path in the virtual resource pool, i.e., a virtual path, is composed of a series of virtual links and NFV nodes between two edge switches.

Infrastructure SDN Controller

With SDN, packet forwarding rules are configured in SDN switches by an infrastructure SDN controller to route traffic flows over a physical path. For virtual link provisioning, the infrastructure SDN controller is responsible for 1) configuring forwarding rules on physical paths associated with each virtual link, and 2) enforcing a traffic splitting ratio among corresponding physical paths for each virtual link.

1.1.2 Tenant Domain

Services

A tenant such as a service provider requests network services in the form of service function chains (SFCs). An SFC is composed of multiple VNFs in a predefined order, to fulfill a composite service with certain processing and transmission resource demands, according to service level agreements (SLAs) negotiated with an infrastructure provider (InP). Each VNF supports a dedicated packet processing functionality. Hence, a network service can provide customized packet processing functionalities in addition to the traditional transmission connectivity to a class of end users. There are two levels of connectivity in an SFC, namely, service-level and infrastructure-level. The service-level connectivity requires that VNFs be chained in a predefined order between the source and destination nodes (fixed at edge switches), to facilitate the E2E service delivery. The service-level connectivity is achieved by mapping an SFC to a virtual path between the source and destination nodes. For two neighboring VNFs in an SFC, packets processed by the upstream VNF are transmitted to the downstream VNF, generating traffic between consecutive VNFs, i.e., inter-VNF subflows. The infrastructure-level connectivity requires that each subflow be routed over at least one physical path, if its upstream and downstream VNFs are not co-located. The infrastructure-level connectivity is achieved by mapping each subflow to a virtual link which is provisioned via the infrastructure SDN controller.

Tenant SDN Controller

The tenant SDN controller configures service-level forwarding rules at edge switches and NFV nodes to guide packets belonging to a flow traversing an SFC (i.e., an SFC flow or a service flow) through a virtual path, thus enabling the service-level connectivity. In the presence of traffic variations, an SFC flow can be rerouted to an alternative virtual path via the tenant SDN controller, according to flow migration decisions made by a central orchestrator.

1.1.3 SDN-NFV Integration

An NFV management and orchestration (MANO) architecture can efficiently manage the life cycle of network functions, services, and their constituent resources in a common NFV infrastructure (NFVI) [3]. The architecture is extended with SDN integration to realize service function chaining [3]. The main functional blocks in the architecture and their interactions with the tenant and infrastructure SDN controllers are introduced as follows.

1) Virtualized infrastructure manager (VIM) is responsible for managing resources in the NFVI. Specifically, the VIM deals with resource virtualization and allocation, and maintains the mapping between the virtual resource pool and physical resource pool. The VIM is also in charge of virtual link provisioning via an infrastructure SDN controller;

2) VNF manager (VNFM) is in charge of the life cycle management of VNFs, including instantiation, configuration, and scaling. In addition to VNFs serving as network service components, the tenant SDN controller is regarded as a VNF;

3) NFV orchestrator (NFVO), which is responsible for central orchestration, contains a resource orchestrator (RO) and a network service orchestrator (NSO). The RO is responsible for orchestrating NFVI resources. For example, it determines the rerouted virtual paths for SFC flows, including both the VNF to NFV node remapping and the consequent subflow to virtual link remapping, as well as the processing and transmission resource scaling for the services. It also determines the virtual link to physical path (re-)mapping, to facilitate dynamic virtual link provisioning. The NSO is responsible for the life cycle management of network services, including service instantiation and dynamic network service capacity scaling. For example, it triggers flow migration and resource scaling requests to the RO when potential QoS violations due to traffic load fluctuations are predicted.

1.2 Research Objectives

With SDN-NFV integration, tenants (e.g., service providers) request network services according to SLAs negotiated with the InP. The resource demands are usually static and estimated from long-term traffic statistics and QoS requirements. The InP customizes multiple network services over a common physical infrastructure, generating service-level network slices (also referred to as virtual networks) [2, 3, 7, 12]. For each service, VNFs are

embedded/placed at NFV nodes, and inter-VNF subflows are routed over physical paths between the corresponding upstream and downstream VNFs. This process is referred to as *SFC embedding* [13–17]. With SFC embedding, a virtual path is established for each SFC flow in the virtual resource pool. SFC embedding at the initial network planning is based on a target traffic load. When data traffic actually enters the network, the traffic load is dynamic and can deviate from the target value, potentially leading to network congestion. There can be traffic dynamics in different time granularities. For example, the traffic statistics (e.g., mean and variance) can be nonstationary and experience significant changes in a coarse time granularity, e.g., larger than 30 minutes, which are usually predictable [18–20]. Within a long time duration with stationary traffic statistics, there are traffic dynamics in small timescales, e.g., around $1ms$, which are usually highly bursty and unpredictable.

To accommodate the traffic dynamics in different time granularities, dynamic resource management among the embedded services is necessary, to ensure efficient and fair operation of the virtual networks with continuous QoS guarantee. Otherwise, congestion can happen, and services can experience performance degradation such as long queueing delay or packet loss due to delay violation. Traditional resource management methods developed for transmission-only networks cannot be directly applied in the service-oriented 5G core networks with the dual-resource environment. The heterogeneity between the processing and transmission resources needs to be captured.

This PhD research is to study the dynamic resource management for the embedded services which share the processing and transmission resources in the network. We focus on three research problems. The first two research problems focus on dynamic resource provisioning for the embedded services to accommodate the large-timescale traffic statistical changes, while the third research problem focuses on dynamic resource scheduling for the embedded services in the presence of the small-timescale traffic dynamics.

In the first research problem, we investigate dynamic flow migration for the embedded services, to avoid QoS degradation due to the mismatch between traffic load and resource availability at the initial virtual path. With flow migration, the SFC flows can migrate to alternative virtual paths with elastic resource allocations. Since many state-dependent VNFs are associated with locally updated states for accurate processing, the states should be transferred to the new location if a VNF is migrated. The state transfers cost transmission resource overhead and incur extra latency. How to achieve a trade-off between load balancing and migration overhead during each flow migration is the focus of the first

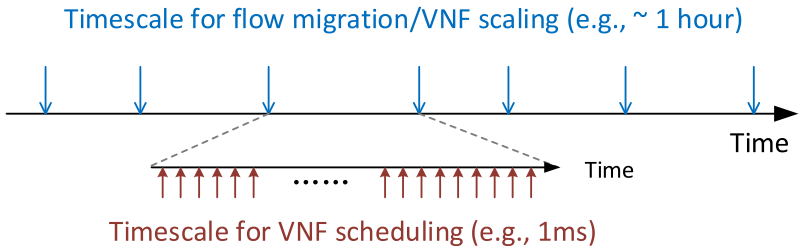


Figure 1.2: An illustration of the timescales for dynamic resource management.

research problem. In the second research problem, we further address two more research questions in flow migration. First, when to trigger resource scaling and possible flow migrations to accommodate large-timescale traffic variations? With traffic fluctuations, a lightly loaded NFV node can become heavily loaded in the future due to increasing background traffic load, and vice versa. How to adapt to traffic patterns and achieve a trade-off between load balancing and migration cost in the long run is the other question. To answer the two questions, we focus on the VNF scaling issue in a local network segment with several candidate NFV nodes, and use a real-world nonstationary traffic trace as traffic input. Although multiple VNFs can be deployed at a common NFV node, how to schedule the central processing unit (CPU) processing resources among them to achieve efficient and fair resource sharing in the presence of small-timescale traffic dynamics should be studied. Therefore, the third research problem is on the VNF scheduling within a sufficiently long time duration, given VNF placement and stationary traffic statistics. The different timescales for flow migration/VNF scaling and VNF scheduling are illustrated in Fig. 1.2.

1.2.1 Dynamic Flow Migration for Embedded Services

During virtual network operation, traffic for each service arrives and fluctuates over time, possibly overloading some NFV nodes and virtual links while underloading some others from time to time. Imbalanced load can create bottlenecks on NFV nodes or virtual links, leading to QoS degradation and possible congestion for the affected services. To avoid QoS violation caused by the load-resource mismatch, we should allow each SFC flow to migrate to an alternative virtual path, to balance the load in the network.

Extensive studies have been done on traffic engineering (TE) to find paths for data delivery from source to destination within link capacity [21, 22]. A cost function, such as a

piece-wise linear increasing and convex function of link utilization, can be used to penalize high link utilization near capacity. The traditional TE ensures that no packets get sent across overloaded links, by minimizing link utilization costs. Service flow migration, i.e., steering SFC flows through alternative NFV nodes and virtual links, is a TE approach for elastic SFC provisioning [23]. Similarly, the maximum loading on NFV nodes can be minimized, to achieve load balancing over processing resources. However, traditional TE methods cannot be directly applied due to the following reasons. First, candidate paths for an SFC flow must traverse through NFV nodes for processing. In traditional TE problems, a flow is a source-destination pair without a predefined sequence of intermediate processing nodes. Second, the transfer of VNF states should be considered, since simply rerouting in-progress flows on a state-dependent VNF to an alternative NFV node leads to state inconsistency, causing processing inaccuracy. Some frameworks such as OpenNF are proposed to solve the state inconsistency problem, by not only migrating packets of the rerouted flow but also transferring the associate VNF states [24–26]. In Co-Scalar [27], parallel state transfer is proposed for an SFC with multiple state-dependent VNFs. Instead of sequentially transferring the states of each VNF, Co-Scalar transfers all VNF states in parallel, thus greatly reducing latency at the cost of transmission resources. Third, the unique properties of processing resources should be considered, such as the VNF-dependent processing density and the switching overhead at each NFV node embedded with multiple VNFs for CPU scheduling.

Dynamic VNF operations, including horizontal scaling, vertical scaling, and migration, are widely employed to provide elastic processing resource provisioning [28]. We consider both vertical scaling and migration under the assumption that the total number of VNF instances is unchanged. Hence, the flow migration problem consists of two joint subproblems, namely, 1) finding the new VNF placement at the NFV nodes and the corresponding new subflow to virtual link remapping, and 2) processing resource scaling for the VNFs and transmission resource scaling for the subflows. There are several studies in recent years on dynamic SFC embedding, in which the time-varying processing and transmission resource demands are assumed known a priori, based on which VNFs are placed at alternative NFV nodes, and inter-VNF subflows are rerouted to different physical paths [29–31]. The QoS requirements are expressed in such a way that the time-varying resource demands should be satisfied without exceeding the resource capacity. In [32], both the resource capacity and delay constraints are taken into consideration, but the load dependent queueing delay is ignored. For QoS provisioning to delay-sensitive services, we consider the queueing delay

and focus on the average E2E delay requirement in the first research problem.

Many VNFs are state-dependent, and states are updated together with packet header or payload processing, to guarantee accurate packet processing. For example, a virtual IDS belonging to an SFC keeps track of pattern matchings for accurate attack detection in subsequent packets. VNF states are stored and updated locally in associate VNFs. During flow migration, the states of the migrated VNFs should be transfered to target NFV nodes for consistency. Hence, VNF state transfers should be taken into consideration in modeling the reconfiguration overhead for flow migration. Existing studies usually model the reconfiguration overhead as a weighted number of reconfigured NFV nodes and physical links [30], or the total revenue loss due to throughput loss within a constant service downtime [31], or the time duration for all state transfers associated with flow migration [33]. One performance metric for migration is the maximum allowable downtime within a certain time duration [34]. Under the assumption that the time interval for flow migration is much larger than state transfer time, we consider to minimize the total transmission resource overhead incurred by state transfers within a maximal tolerable service downtime in one service interruption. If the VNF states have the same size, minimizing the total transmission resource overhead under parallel state transfer is equivalent to minimizing the total number of state transfers, i.e., minimizing the modification to the current VNF placement at the NFV nodes. In this case, the loads on different NFV nodes can be rather imbalanced, with some heavily loaded and some lightly loaded, which can possibly result in more migrations in the future. Therefore, we have another goal to minimize the maximum NFV node loading factor to achieve load balancing. The two goals can conflict with each other. For example, a pure load balancing solution may result in frequent VNF migrations. With the consideration that not every two NFV nodes are directly connected by virtual links, the number of extra virtual links required for flow rerouting should also be minimized to reduce signaling overhead. Therefore, we should jointly consider the three objectives in the flow migration problem.

In Problem I, we aim to develop a delay-aware flow migration scheme for the embedded services, to guarantee the average E2E delay performance of each service, while addressing the trade-off between load balancing and migration overhead, under resource capacity constraints and maximal tolerable service downtime constraints, with the consideration of the service chaining requirements and the unique properties of processing resources such as the VNF-dependent processing density and the switching overhead.

1.2.2 Dynamic Resource Scaling for VNF over Nonstationary Traffic

The processing resource demand of a VNF is dependent on both the statistics of traffic arrivals and the QoS requirement. With changes in traffic statistics, the processing resource demand of a VNF varies for a certain QoS requirement. In Problem II, we consider a more stringent delay requirement than the average delay requirement. Suppose that the E2E delay bound of a service is decomposed into per-hop delay bounds at each VNF. Then, a probabilistic delay requirement at a VNF requires that the probability of packet processing (including queueing) delay at a VNF exceeding a certain delay bound should not be beyond an upper limit. Existing studies usually assume prior knowledge about the time-varying resource demands or predict the future resource demands based on historical resource demand information [18, 19, 31, 35]. The average traffic rate in a certain time duration is usually considered as the resource demand [18, 19]. However, resource allocation/scaling according to the average traffic rate is not sufficient to satisfy a stringent probabilistic delay requirement. In Problem I, we determine the processing resource demand at different VNFs in a service jointly to satisfy a less stringent average E2E delay requirement, under the assumption of Poisson traffic model. In reality, the real-world traffic is more bursty than Poisson traffic. Also, a real-world resource demand trace with inherent QoS guarantee is difficult to obtain. Instead, a traffic trace with packet arrival information is usually available [20]. Therefore, a resource demand prediction scheme is required, to predict the time-varying QoS-aware resource demands following the traffic statistical changes in an available packet arrival traffic trace. Then, VNF scaling decisions can be made, to scale up/down the amount of resources allocated to the VNFs according to the predicted resource demands and to update the placement of VNFs among several candidate NFV nodes. There can be overlapping among the sets of candidate NFV nodes for different VNFs. To focus on traffic analysis and resource demand prediction in Problem II, we consider one VNF in a neighborhood with several candidate NFV nodes, and treat the dynamics of other VNFs as background traffic at the NFV nodes. The dynamics of other VNFs are attributed to dynamics in both their traffic arrivals and scaling decisions.

There are several existing studies on dynamic VNF placement and traffic routing, based on decisions made in a proactive or reactive manner at consecutive non-overlapping decision epochs of equal length, e.g., 30 minutes [18, 19, 31]. In Problem I, we assume Poisson traffic model with changing packet arrival rates across different time intervals, but how

to determine the interval length is not addressed. The selection of interval/epoch length is difficult and usually based on experience. If the decision epoch is too long, traffic burstiness in different time granularities within an epoch cannot be captured, resulting in challenges for continuous QoS guarantee; if the decision epoch is too short, decisions are made frequently, possibly resulting in unnecessary expensive VNF migrations for temporal short traffic bursts. A better way is to adopt adaptive epoch length according to changes in traffic statistics (e.g., mean and variance) and resource demands, since the real-world traffic usually exhibits nonstationary traffic characteristics across intervals with uncertain time durations. Several change point detection algorithms, either retrospective or online, have been developed for detecting structural breaks in a nonstationary time series [36–38]. Online algorithms provide inference about change points as each data sample arrives, which is more appropriate for detecting traffic statistical changes, based on which VNF scaling decisions can be made reactively without a significant latency [37, 38]. Under the assumption that a nonstationary traffic trace can be partitioned into consecutive stationary traffic segments with unknown change points, the decision epochs with variable lengths are to be identified based on change point detection. Each stationary traffic segment corresponds to one decision epoch. Traffic arrivals of a VNF are from a service-level flow which is an aggregation of traffic flows of different users. In core and backbone networks, the aggregation level is high, which makes Gaussian traffic approximation work well beyond a timescale of around 100 ms [39]. Gaussianity of a certain distribution can be checked by quantile-quantile (QQ) plot versus a standard Gaussian distribution [40]. Fractional Brownian motion (fBm) is a Gaussian process with properties such as self-similarity and long-range dependence (LRD) which comply with the properties of real-world network traffic [39, 41]. Hence, we adopt the fBm traffic model, based on which the characteristic traffic parameters of each stationary traffic segment are learned, and the corresponding resource demands are predicted.

At each decision epoch, a VNF scaling decision is made, which possibly requires VNF migrations. To address the trade-off between load balancing and migration cost reduction, we jointly consider the two objectives, by jointly minimizing the migration cost and the maximum resource loading factor among all candidate NFV nodes. Moreover, there is a trade-off between cost minimizations in the short term and in the long run. When a VNF migration is required, the VNF is migrated to the current most lightly loaded NFV node for cost minimization in the current decision epoch. However, a lightly loaded NFV node can become heavily loaded in the future due to increasing background resource usage, resulting

in further migrations to avoid performance degradation. In contrast, for cost minimization in the long run, the VNF should be migrated to an NFV node which is expected not to be heavily loaded in the current and successive decision epochs. Reinforcement learning (RL) provides an approach for long-run cost minimization, with the ability to capture inherent patterns in network dynamics and to make intelligent decisions accordingly [19, 42–47].

In Problem II, the objective is to develop a learning-based dynamic VNF scaling scheme, to adaptively trigger and perform VNF migration and resource scaling decisions based on detected traffic statistical changes in a real-world nonstationary traffic trace, while satisfying the probabilistic delay requirement. For simplicity, we consider one VNF in a local network segment with several NFV nodes. A traffic parameter learning scheme is to be developed based on change point detection, to learn traffic parameters of each detected stationary traffic segment in a real-world nonstationary traffic trace, based on which resource demand prediction can be performed.

1.2.3 Delay-Aware VNF Scheduling For Network Utility Maximization

In Problems I and II, the VNF placement and resource allocation are adjusted based on traffic statistical changes in large time granularities. Within a sufficiently long time duration, e.g., an hour, with stationary traffic statistics for each service, the VNF placement at the NFV nodes remains unchanged, and the CPU processing resource budgets at the NFV nodes for the services are fixed. Although an NFV node can hold multiple VNFs with resource sharing among each other, we assume that at most one VNF can be scheduled for packet processing and occupy the CPU resources at an NFV node at a time instant. Hence, an efficient CPU processing resource scheduling scheme executed in small timescale is required. In Problem III, we want to determine which VNF (belonging to different services) to schedule at each NFV node and how many packets to be processed from the scheduled VNF, while achieving efficient and fair resource sharing among services in the presence of small-timescale traffic dynamics.

The packet-level transmission resource scheduling schemes have been extensively investigated, such as the generalized processor sharing (GPS) scheme in which the resources are infinitely divisible under the assumption of infinitely small time slots [12, 48]. Such an assumption is acceptable for packet-level transmission resource scheduling in a very tiny

timescale, e.g., ns to μs. For multiple traffic flows sharing a transmission link, each flow is guaranteed a minimum transmission rate proportional to the assigned weight, with multiplexing gain among flows. In the worst case without multiplexing gain, the traffic flows enjoy as if dedicated transmission links which can be scheduled simultaneously, each supporting a minimum transmission rate for one flow. However, the CPU processing resource scheduling in the NFV environment is on the software process level. Each VNF corresponds to a software process. Once a VNF is scheduled, it occupies the CPU for a certain time duration and a batch of packets are processed. The time granularity for process scheduling should not be too small, to avoid frequent switching overhead between different scheduled VNF processes, such as the CPU scheduling overhead for selecting the next process to run and the context switching time overhead for saving and loading contexts [49]. The minimum time quantum in some *state-of-the-art* operating system (OS) process schedulers is in the $100\mu s$ to ms timescale, such as $100\mu s$ for the completely fair scheduler (CFS) and $1ms$ for the real time scheduler (RLS) [50]. Some OS schedulers developed for the NFV environment can support a smaller time quantum such as $10\mu s$, but they are still in the initial development stage [51]. Hence, the assumption of infinitely divisibility is unrealistic for CPU processing resources, and the GPS scheme cannot be directly applied to CPU processing resource scheduling among co-located VNFs.

Some leading applications in the 5G and beyond era, e.g., virtual reality (VR) services, have extreme requirements in the rate-latency-reliability space [52]. Such a strict QoS requirement can be expressed as a high timely throughput requirement. Only packets delivered within a hard E2E deadline are counted in the timely throughput [53–55]. Any packets with E2E delay violation are expired and should be dropped. The packet dropping ratio should be extremely low to guarantee a high reliability. We use *throughput* for such a deadline-constrained service to represent *timely throughput*. With a *state-of-the-art* time quantum in the timescale of $100\mu s$ to ms, the sequences for scheduling the VNFs belonging to different services with resource sharing among each other have a significant impact on the QoS performance of each service. For example, if we use a time quantum (or time slot) of $1ms$ for VNF scheduling, all packets of a service with an E2E deadline of $10ms$ will be expired if the VNFs are not scheduled within 10 time slots in an extreme case. How to coordinate the VNF scheduling for different services sharing a group of NFV nodes with a guarantee for the strict QoS requirements becomes more difficult as the resource sharing among different services is more complicated. The coordination of VNF scheduling within a service is also important, since the E2E QoS performance depends on scheduling of all

VNFs in the chain. If a VNF experiences congestion, it can help to relieve the congestion situation by not scheduling the upstream VNF temporarily.

There are existing studies on delay-aware VNF scheduling, which focus on reducing the maximum E2E delay for the sequential processing of a given traffic block at different VNFs in each service [46, 56–59]. An inherent assumption is that the processing for a traffic block at different VNFs in a chain has no time overlapping, inferring that the first packet in the traffic block has to wait until the last packet finishes processing, which is inefficient for services with a strict per-packet E2E deadline especially if the traffic block size is big. Hence, a QoS-aware and chain-aware VNF scheduling scheme using a *state-of-the-art* time quantum is required, to satisfy the strict throughput requirement of each deadline-constrained service.

Moreover, the traffic dynamics in small time granularities such as $1ms$ are usually highly bursty and unpredictable. With the unavailability of small-timescale traffic statistics, a potential chain-aware VNF scheduling approach is an adaptation of the classical backpressure algorithm which was originally developed for transmission resources and has been proved to be throughput-optimal for delay-insensitive flows [60]. With the potential approach, the differential backlogs in number of required CPU cycles is used as the VNF scheduling weight [14, 61, 62]. However, for delay-sensitive services with high throughput requirements, such a simple adaptation cannot be directly applied. Hence, we aim to develop a delay-aware backpressure-based VNF scheduling policy by incorporating packet delay and throughput performance into the VNF scheduling.

In Problem III, the objective is to develop a delay-aware VNF scheduling policy to coordinate the VNF scheduling for different deadline-constrained services, which can be executed in a realistic *state-of-the-art* VNF scheduling time quantum and at the same time satisfies the high throughput requirements in the presence of bursty and unpredictable small-timescale traffic dynamics.

1.3 Research Contributions

In this section, we present the research contributions for each research problem.

1.3.1 Dynamic Flow Migration for Embedded Services

In Chapter 2, we develop a delay-aware flow migration model for multiple delay-sensitive services in a processing resource limited network, to guarantee the average E2E delay isolation among services within maximal tolerable service downtime, while addressing the trade-off between load balancing and migration overhead. A multi-objective mixed integer optimization problem is formulated, which minimizes a weighted summation of the maximum NFV node loading factor, the total transmission resource overhead incurred by state transfers, and the number of extra virtual links for flow rerouting. For delay awareness, under the assumption of Poisson traffic model and prior knowledge of time-varying packet arrival rates, the average E2E delay requirements are included in constraints based on M/M/1 queue based delay modeling. Under the assumption of sufficient transmission resources, we ignore the delay on virtual links. The processing resource constraints are incorporated with the consideration of VNF dependent processing density and switching overhead. Due to several quadratic constraints, an optimal solution to the problem is difficult to obtain using solvers such as Gurobi [63]. We transform the original problem into a tractable mixed integer quadratically constrained programming (MIQCP) problem. Although the two problems are not equivalent, it is proved that there is a zero optimality gap between them. Given an MIQCP optimum, the optimum of the original problem is obtained through a proposed mapping algorithm. The MIQCP transformation, together with the mapping algorithm, gives an optimal solution, but time complexity is high due to NP-hardness of the MIQCP problem. Therefore, a low-complexity heuristic algorithm based on redistribution of hop delay bounds is proposed to obtain a sub-optimal solution to the original problem [64–66].

1.3.2 Dynamic Resource Scaling for VNF over Nonstationary traffic

In Chapter 3, we use machine learning tools to develop an adaptive VNF scaling mechanism with a real-world nonstationary traffic trace as input. We first develop a two-timescale resource demand prediction framework, based on a change point detection scheme using traffic samples in medium timescale (e.g., $20s$) and a traffic parameter learning scheme using traffic samples in a smaller timescale (e.g., $100ms$). Then, we propose a deep-$\mathcal{Q}$ learning approach for VNF migration decision, which learns the traffic pattern and

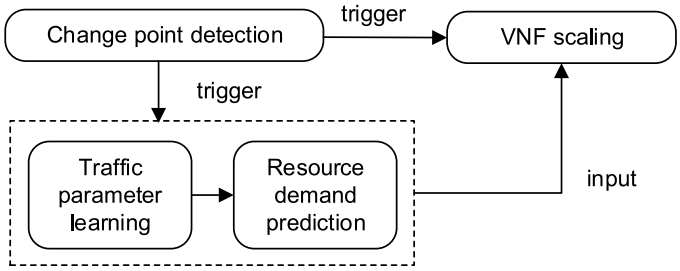


Figure 1.3: Workflow of dynamic VNF scaling over nonstationary traffic.

addresses the trade-off between load balancing and migration cost in the long run. The new contributions are summarized as follows, with a workflow given in Fig. 1.3 [67].

- We first use a Bayesian online change point detection (BOCPD) algorithm to detect statistical changes in mean and variance of the medium-timescale (e.g., $20s$) time series. The algorithm provides online estimation of a probability distribution of current run length and the most probable mean and variance at each medium-timescale traffic sample. A run is defined as a traffic segment with the same statistics. Then, we use a threshold-based policy to identify deterministic change points (boundaries) between consecutive stationary traffic segments. The employed machine learning tool is Bayesian conjugate analysis.

- After a new change point is detected, the traffic parameters of the upcoming stationary traffic segment should be learned. Since the BOCPD algorithm is statistical, it results in a latency between the real change points and the detected change points. We exploit the latency for a look-back traffic parameter learning scheme. A number of small-timescale (e.g., $100ms$) traffic samples before the detected change point are collected for traffic model regression. Since the core network traffic has a high aggregation level, the Gaussian traffic approximation works well beyond a timescale of around $100ms$. Hence, we adopt the fractional Brownian motion (fBm) traffic model for each stationary traffic segment to incorporate Gaussianity and other properties of real-world core network traffic such as self-similarity and Long-range dependence (LRD). Then, the fBm traffic parameters can be learned through training a Gaussian process regression (GPR) model with a selected fBm covariance (kernel) function. Afterwards, the resource demand of the upcoming stationary traffic segment for a required QoS performance is calculated using empirical models.

- Change point detection provides a triggering signal for VNF scaling. Hence, the length of a VNF scaling decision epoch is varying and depends on change point detection. With the detected change points and predicted resource demands, a VNF migration problem is formulated as a Markov decision process (MDP) with variable-length decision epochs, to minimize the overall cost integrating imbalanced loading, migration cost, and resource overloading penalty in the long run. A deep $\mathcal{Q}$-learning algorithm with penalty-aware prioritized experience replay is proposed to solve the MDP, with performance gains in terms of both cost and training loss reduction compared with benchmark algorithms.

1.3.3 Delay-Aware VNF Scheduling For Network Utility Maximization

In Chapter 4, a delay-aware VNF scheduling problem is studied in the presence of bursty and unpredictable small-timescale traffic dynamics, to coordinate the VNF scheduling for different deadline-constrained services with high throughput requirements, while using a realistic *state-of-the-art* VNF scheduling time quantum. The main contributions are summarized as follows.

- We use a packet delay aware queueing model for each service, by introducing virtual packet processing queues augmented with packet delay information at each VNF, which is the foundation for developing a delay-aware VNF scheduling algorithm.
- For efficient and fair utilization of the allocated resources at the NFV nodes, the VNF scheduling problem is formulated as a stochastic offline problem which maximizes a total network utility with proportional fairness among services, while stabilizing all the VNF packet processing queues and satisfying the throughput constraints of all the deadline-constrained services. The utility function is a strictly increasing and concave function of throughput. The stochastic offline problem is transformed into an online problem by decoupling the VNF scheduling decisions over time slots with the Lyapunov optimization technique, based on which an online VNF scheduling algorithm is derived [14, 68]. Distributed VNF scheduling decisions are made at each NFV node for each time slot, based on the observed local status, such as the delay-aware virtual processing queue lengths, the packet arrivals, and the service QoS performance indicated by the lengths of some specially designed virtual queues.

- The VNF scheduling algorithm is developed under the assumption that a packet processed at a VNF has to wait until the beginning of a new time slot for further processing at downstream VNFs. Without such an assumption, a packet is allowed to be processed by several consecutive VNFs during one time slot, which is referred to as packet rushing. We also propose a modified VNF scheduling algorithm with the consideration of packet rushing, and correct the queue length updates based on a packet rushing analysis for each service given the VNF scheduling status and resource availability.

1.4 Thesis Outline

The rest of this thesis is organized as follows. The three research problems and solutions are presented in Chapters 2, 3, and 4, respectively.

In Chapter 2, Section 2.1 presents the system model under the consideration of Problem I. The delay-aware flow migration problem is formulated in Section 2.2. Section 2.3 presents the MIQCP problem transformation, and derives the optimality gap between the transformed problem and the original problem. A low-complexity heuristic algorithm is proposed in Section 2.4. Performance evaluation for both the MIQCP and heuristic solutions is presented in Section 2.5, and a summary for this chapter is presented in Section 2.6. In Chapter 3, Section 3.1 presents the system model under the consideration of Problem II. The change-point-driven traffic parameter learning and resource demand prediction schemes are proposed in Section 3.2. Section 3.3 presents the MDP formulation and the penalty-aware deep Q-learning algorithm. Performance evaluation is given in Section 3.4, and a summary for this chapter is presented in Section 3.5. In Chapter 4, Section 4.1 presents the system model under the consideration of Problem III. Section 4.2 presents the VNF scheduling problem formulation and transformation, based on which an online distributed VNF scheduling algorithm is derived in Section 4.3. Section 4.4 presents the modified VNF scheduling algorithm with packet rushing analysis. Performance evaluation is given in Section 4.5, and a summary for this chapter is given in Section 4.6. Finally, concluding remarks are drawn and future research directions are discussed in Chapter 5.

Chapter 2

Dynamic Flow Migration for Embedded Services

In this chaper, a dynamic flow migration problem for embedded services is studied, to meet average E2E delay requirements with time-varying traffic. A multi-objective mixed integer optimization problem is formulated, addressing the trade-off between load balancing and reconfiguration overhead. The problem is transformed to a tractable MIQCP problem. It is proved that there is no optimality gap between the two problems; hence, we can obtain the optimum of the original problem by solving the MIQCP problem with some post-processing. To reduce time complexity, a heuristic algorithm based on redistribution of hop delay bounds is proposed to find an efficient solution. Numerical results are presented to demonstrate the aforementioned trade-off, the benefit from flow migration in terms of E2E delay guarantee, as well as the effectiveness and efficiency of the heuristic solution.

2.1 System Model

A time-slotted system is considered, with integer k denoting the k-th time interval over which a flow migration plan remains unchanged.

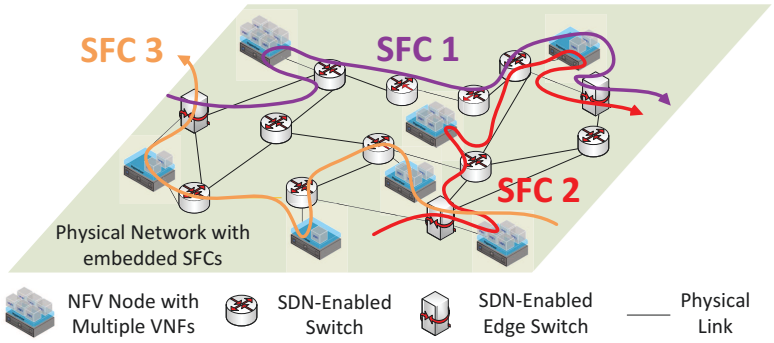


Figure 2.1: A physical network with embedded SFCs.

2.1.1 Services

Let $\mathcal{R}$ denote the set of embedded services. A service, $r \in \mathcal{R}$, is represented in the form of SFC. It originates from source node $\mathtt{n}_1^{(r)}$ and traverses through H_r VNFs in sequence towards destination node $\mathtt{n}_2^{(r)}$, with average E2E delay requirement D_r, maximal tolerable downtime Ω_r in one service interruption, average packet size b_r in bit, and time-varying traffic rate $\lambda^{(r)}(k)$ in packet/s. Under the assumption that flow migration is not frequent and the time interval is sufficiently large, traffic arrival of SFC $r \in \mathcal{R}$ during time interval k is modeled as Poisson with rate $\lambda^{(r)}(k)$ packet/s. Under the assumption that the time interval is much larger than $\max_{r\in\mathcal{R}} \Omega_r$, the experienced service downtime is much shorter than stable service operation time for any service. Let $\mathcal{H}_r = \{1, \cdots, H_r\}$, and denote the h-th ($h \in \mathcal{H}_r$) VNF in SFC r as $V_h^{(r)}$. Let $V_0^{(r)}$ and $V_{H_r+1}^{(r)}$ be dummy VNFs in SFC r, locating at source node $\mathtt{n}_1^{(r)}$ and destination node $\mathtt{n}_2^{(r)}$ respectively. Let $\mathcal{V}$ be a set containing all VNFs belonging to different SFCs, with $(r, h) \in \mathcal{V}$ denoting the h-th VNF in SFC r. Let $\mathcal{A}$ be a set of edge switches hosting all dummy VNFs. The h-th ($h \in \{0\} \cup \mathcal{H}_r$) inter-VNF subflow in SFC r, i.e., the subflow between upstream (dummy) VNF $V_h^{(r)}$ and downstream (dummy) VNF $V_{h+1}^{(r)}$, is denoted as $Y_h^{(r)}$.

2.1.2 Abstraction of Virtual Resource Pool

Fig. 2.1 shows a physical network with three embedded SFCs in single-path routing. A virtual resource pool is abstracted from the physical network with embedded SFCs, represented as a directed graph $G = \{\mathcal{N} \cup \mathcal{A}, \mathcal{E}\}$, where $\mathcal{N}$ is a set of all NFV nodes, $\mathcal{A}$ is a set of

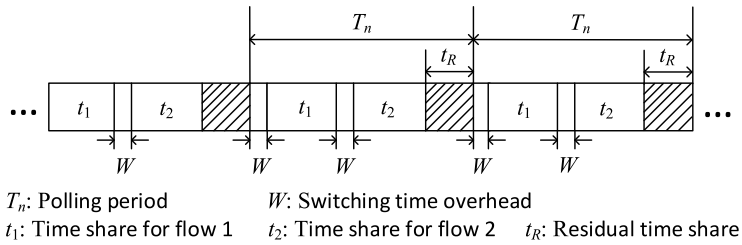


Figure 2.2: A CPU polling scheme with two flows.

edge switches hosting dummy VNFs, and $\mathcal{E}$ is a set of virtual links. Here, we assume that each NFV node can hold all types of VNFs for simplicity. In reality, the supported VNF types at each NFV node can be differentiated, and the placement for a VNF of a certain type should be restricted to the NFV nodes which support the same type. For virtual link $e \in \mathcal{E}$, we use binary parameters, $\{I_n^{e,1}\}$ and $\{I_n^{e,2}\}$, to describe its location and direction, with $I_n^{e,1}= 1$ if $n \in \mathcal{N} \cup \mathcal{A}$ is its starting point and $I_n^{e,2}= 1$ if $n \in \mathcal{N} \cup \mathcal{A}$ is its ending point. It is possible that G is not fully connected. Assume that there are sufficient transmission resources in the physical network. We can increase resources on existing virtual links, and find paths with enough resources for extra virtual links. Hence, we consider a processing resource limited virtual resource pool.

2.1.3 Processing Resource Sharing

The processing resource capacity C_n of NFV node $n \in \mathcal{N}$ is its maximum supporting CPU processing rate in cycle/s. For one packet/s of processing rate, the CPU resource demand on a certain NFV node depends on many factors, including the packet size, the type of function, the packet I/O scheme, and the virtualization technology [69–71]. We summarize all the factors into two categories: VNF dependent, and NFV node dependent. We define processing density of VNF $V_h^{(r)}$ at NFV node n as P_n^{rh} (in cycle/bit), which is the CPU resource demand (in cycle/s) of VNF $V_h^{(r)}$ at NFV node n for one bit/s of processing rate. Accordingly, $P_n^{rh} b_r$ is the processing density of VNF $V_h^{(r)}$ at NFV node n in cycle/packet. A CPU polling scheme is employed for resource sharing among multiple VNFs, as illustrated in Fig. 2.2, in which two VNF processing queues are polled for service. Each queue gets a portion of CPU resources which is linear with its allocated CPU time share in a polling period. The polling scheme introduces CPU scheduling overhead and multi-task context

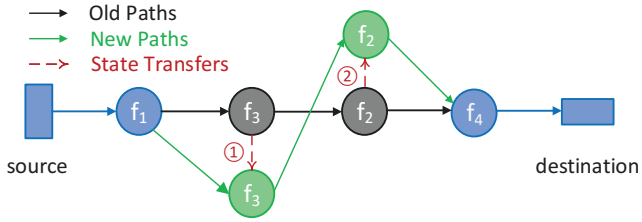


Figure 2.3: An illustration of flow migration and state transfer.

switching overhead, due to extra CPU time spent on determining the next VNF process to run and on saving and loading contexts between every two consecutive tasks [49]. Here, processing packets from a certain VNF processing queue is a task. The total time overhead for switching between VNFs are referred to as the switching time overhead. The polling period T_n and the switching time overhead $\mathtt{W}$ in NFV node n are constant. Under the assumption that resources are infinitely divisible, GPS is a benchmark resource allocation scheme to achieve QoS isolation and multiplexing gain among flows [12, 72]. Without such an assumption for CPU processing resources, we assume that there exists a practical VNF scheduling algorithm (to be studied in Problem III) which achieves comparable performance with GPS. Then, the two flows are guaranteed minimum processing rates (in packet/s) of $S_1 = \frac{t_1 \mathtt{C}_n}{T_n P_n^1 b_1}$ and $S_2 = \frac{t_2 \mathtt{C}_n}{T_n P_n^2 b_2}$ respectively, where $t_1 + t_2 + 2\mathtt{W} = T_n - t_R$, and t_R is the residual time share in a polling period. Define loading factor of NFV node n, denoted by η_n, as the percentage of allocated time shares plus switching time overhead in a polling period of NFV node n. Assume that the switching time overhead at NFV node n linear increases with the number of VNFs placed at the NFV node if the number of VNFs is larger than 1.

2.1.4 Reconfiguration Overhead

When an SFC flow migrates at a state-dependent VNF, the VNF is remapped to an alternative NFV node, with the associate states transferred to the target NFV node. Fig. 2.3 illustrates the flow migration and associate state transfers, where two VNFs are remapped, and two state transfers are triggered correspondingly. Packet processing is halted during state transfer, incurring a service downtime. Let $U_h^{(r)}(k)$ (in bit) be the time-varying state size of VNF $V_h^{(r)}$, whose value at time interval k is monitored by the SDN controller. For a state transfer at VNF $V_h^{(r)}$, $U_h^{(r)}(k)$ is the product of state transfer delay and consumed transmission resources (in bit/s) [33]. For a remapped SFC with multiple state transfers,

we use parallel state transfer in data plane in which all state transfers can take place simultaneously [27]. Then, the service downtime, which is the maximum state transfer delay along the E2E path, is much less than that of sequential state transfer, at the cost of transmission resources. Within a maximal tolerable service downtime, the total transmission resource overhead incurred by state transfers should be minimized. We assume that the transmission resource overhead for each state transfer is not less than $B_{min} = \frac{min(U_h^{(r)}(k))}{max(\Omega_r)}$.

For a subflow, if its upsteam or downstream VNF migrates to an alternative NFV node, the subflow should be remapped to an alternative virtual link accordingly. After a successful remapping, transmission resources allocated to the subflow are released from the old virtual link. However, it is possible that NFV nodes in the virtual resource pool are not fully connected. Assume that the infrastructure SDN controller can find physical paths with enough resources for an extra virtual link. Forwarding rule configuration along the physical paths incurs signaling overhead between the SDN controller and SDN-enabled switches.

Hence, the reconfiguration overhead due to flow migration is described in two parts: the total transmission resource overhead incurred by state transfers, and the total signaling overhead for configuring extra virtual links required for flow rerouting. The latter is assumed to be linear with the total number of extra virtual links required for flow rerouting.

2.2 Problem Formulation

Consider a processing resource limited virtual resource pool. Assume that packet processing time at an NFV node for an SFC is exponentially distributed [73, 74]. During time interval k, traffic arrival of SFC $r \in \mathcal{R}$ is Poisson with rate $\lambda^{(r)}(k)$ packet/s. The rate can be predicted at the end of time interval $(k-1)$ based on measurements and historical information [19, 23]. A delay-aware flow migration problem is to 1) find the remapping between VNFs and NFV nodes in time interval k, based on the old mapping in time interval $(k-1)$, and 2) scale the processing resources allocated to VNFs, to satisfy average E2E delay requirements without violating processing resource constraints. The objective function for time interval k, $O(k)$, is to achieve load balancing among NFV nodes, with minimal reconfiguration overhead, given by

$$O(k) = \omega_1 \eta(k) + \omega_2 \sum_{(r,h)\in\mathcal{V}} \sum_{n,n'\in\mathcal{N}} \frac{B^{rh}_{n,n'}(k)}{B_{min}} + \omega_3 \sum_{r\in\mathcal{R}} \sum_{h\in\{0\}\cup\mathcal{H}_r} \sum_{n,n'\in\mathcal{N}\cup\mathcal{A}} y^{rh}_{nn'}(k). \tag{2.1}$$

In objective function (2.1), there are several decision variables for time interval k: 1) continuous variable $\eta(k) \in [0,1]$ for maximum loading factor among all NFV nodes during interval k; 2) nonnegative continuous variable set $\boldsymbol{B}(k) = \{B^{rh}_{n,n'}(k)\}$, with $B^{rh}_{n,n'}(k)$ being the transmission resource overhead to transfer the state of VNF $V^{(r)}_h$ from NFV node $n \in \mathcal{N}$ during interval $(k-1)$ to NFV node $n' \in \mathcal{N}$ during interval k; 3) binary variable set $\boldsymbol{y}(k) = \{y^{rh}_{nn'}(k)\}$, with $y^{rh}_{nn'}(k) = 1$ if subflow $Y^{(r)}_h$ is mapped to an extra virtual link between $n, n' \in \mathcal{N} \cup \mathcal{A}$ during interval k, and $y^{rh}_{nn'}(k) = 0$ otherwise. Note that ω_1, ω_2, ω_3 are tunable weights to control the priority of the three components, with $\omega_1 + \omega_2 + \omega_3 = 1$. In the right hand side of (2.1), the first term is the cost for imbalanced loading among NFV nodes since minimizing $\eta(k)$ achieves load balancing among all the NFV nodes, the second term is the cost for the overall normalized transmission resource overhead due to state transfers with a normalization ratio of $\frac{1}{B_{min}}$, the third term is the cost for extra virtual links required by flow rerouting. The normalization makes the three components in objective function (2.1) comparable, based on which ω_1, ω_2, ω_3 can be selected on the same order of magnitude. A component in (2.1) is ignored if the corresponding weight is set to 0. If all weights are positive, all components in (2.1) are jointly optimized.

In terms of constraints, we start from node mapping constraints. Define binary decision variable set $\mathbf{x}(k) = \{\mathrm{x}^{rh}_n(k)\}$ for interval k, with $\mathrm{x}^{rh}_n(k) = 1$ if (dummy) VNF $V^{(r)}_h$ is mapped to node $n \in \mathcal{N} \cup \mathcal{A}$ during interval k, and $\mathrm{x}^{rh}_n(k) = 0$ otherwise. As VNF $V^{(r)}_h$ should be mapped to exactly one NFV node in $\mathcal{N}$, we have

$$\sum_{n\in\mathcal{N}} \mathrm{x}^{rh}_n(k) = 1, \quad \forall (r,h) \in \mathcal{V}. \tag{2.2}$$

For dummy VNFs, i.e., source and destination nodes, their physical locations are fixed and confined by

$$\mathrm{x}^{r0}_{\mathtt{n}^{(r)}_1}(k) = 1, \qquad r \in \mathcal{R} \tag{2.3a}$$

$$\mathrm{x}^{r0}_n(k) = 0, \qquad r \in \mathcal{R}, n \in \mathcal{N} \cup \mathcal{A} \backslash \mathtt{n}^{(r)}_1 \tag{2.3b}$$

$$\mathrm{x}^{r(H_r+1)}_{\mathtt{n}^{(r)}_2}(k) = 1, \quad r \in \mathcal{R} \tag{2.3c}$$

$$\mathrm{x}^{r(H_r+1)}_n(k) = 0, \quad r \in \mathcal{R}, n \in \mathcal{N} \cup \mathcal{A} \backslash \mathtt{n}^{(r)}_2. \tag{2.3d}$$

The next constraints are related to transmission resource overhead for state transfer. From interval $(k-1)$ to interval k, the set representing VNF to NFV node mapping changes from $\mathbf{x}(k-1) = \{\mathrm{x}_n^{rh}(k-1)\}$ to $\mathbf{x}(k) = \{\mathrm{x}_n^{rh}(k)\}$, where $\mathbf{x}(k-1)$ is known in interval k. We introduce binary decision variable set $\boldsymbol{g}(k) = \{g_{n,n'}^{rh}(k)\}$ for interval k, with $g_{n,n'}^{rh}(k) = 1$ if the mapped NFV node for VNF $V_h^{(r)}$ changes from NFV node $n \in \mathcal{N}$ during interval $(k-1)$ to NFV node $n' \in \mathcal{N}$ during interval k, and $g_{n,n'}^{rh}(k) = 0$ otherwise. Hence, there is a relationship constraint among $\{g_{n,n'}^{rh}(k)\}$ $(n \neq n')$, $\{\mathrm{x}_n^{rh}(k-1)\}$ and $\{\mathrm{x}_{n'}^{rh}(k)\}$, given by

$$g_{n,n'}^{rh}(k) = \mathrm{x}_n^{rh}(k-1)\mathrm{x}_{n'}^{rh}(k), \quad \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N}, \forall n' \in \mathcal{N} \backslash \{n\}. \tag{2.4}$$

Also, we have

$$g_{n,n}^{rh}(k) = 0, \quad \forall (r,h) \in \mathcal{V},\ \forall n \in \mathcal{N}. \tag{2.5}$$

According to the definition of $B_{n,n'}^{rh}(k)$, we have

$$0 \leq B_{n,n'}^{rh}(k) \leq g_{n,n'}^{rh}(k)\mathbb{M}, \quad \forall (r,h) \in \mathcal{V},\ \forall n, n' \in \mathcal{N} \tag{2.6}$$

where $\mathbb{M}$ is a big-$\mathbb{M}$ constant to guarantee that $B_{n,n'}^{rh}(k) = 0$ if $g_{n,n'}^{rh}(k) = 0$. Let $\boldsymbol{u}(k) = \{u_{n,n'}^{rh}(k)\}$ be a positive continuous decision variable set for interval k, with $u_{n,n'}^{rh}(k)$ denoting the (dummy) delay to transfer state of VNF $V_h^{(r)}$ from NFV node $n \in \mathcal{N}$ during interval $(k-1)$ to NFV node $n' \in \mathcal{N}$ during interval k. It follows that

$$u_{n,n'}^{rh}(k) = \frac{U_h^{(r)}(k)}{B_{n,n'}^{rh}(k) + \epsilon_1}, \quad \forall (r,h) \in \mathcal{V},\ \forall n, n' \in \mathcal{N} \tag{2.7}$$

where $0 < \epsilon_1 \ll 1$ is a constant to avoid $u_{n,n'}^{rh}(k)$ being undetermined, and $u_{n,n'}^{rh}(k)$ is a dummy delay only if $g_{n,n'}^{rh}(k) = 0$. Moreover, $u_{n,n'}^{rh}(k)$ has an upper bound

$$0 < u_{n,n'}^{rh}(k) \leq g_{n,n'}^{rh}(k)\Omega_r + \left[1 - g_{n,n'}^{rh}(k)\right] \frac{U_h^{(r)}(k)}{\epsilon_1}, \quad \forall (r,h) \in \mathcal{V},\ \forall n, n' \in \mathcal{N}. \tag{2.8}$$

If $g_{n,n'}^{rh}(k) = 1$, the upper bound is the corresponding maximal tolerable service downtime Ω_r; otherwise, it is $\frac{U_h^{(r)}(k)}{\epsilon_1}$.

For constraints related to processing resource scaling, let $\boldsymbol{S}(k) = \{S_n^{rh}(k)\}$ be a non-negative continuous decision variable set for interval k, with $S_n^{rh}(k)$ being the processing

rate in packet/s allocated to VNF $V_h^{(r)}$ by NFV node $n \in \mathcal{N}$ during interval k. It should be lower bounded by $\mathrm{x}_n^{rh}(k)\lambda^{(r)}(k)$ due to the queue stability requirement and upper bounded by $\frac{\mathrm{x}_n^{rh}(k)\mathsf{C}_n}{P_n^{rh}b_r}$ due to the limited processing capacity, given by

$$\mathrm{x}_n^{rh}(k)\lambda^{(r)}(k) \leq S_n^{rh}(k) \leq \frac{\mathrm{x}_n^{rh}(k)\mathsf{C}_n}{P_n^{rh}\, b_r}, \quad \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N}. \tag{2.9}$$

Let $\mathrm{w}_n(k)$ be a binary decision variable with $\mathrm{w}_n(k) = 1$ if switching happens at NFV node n during interval k, i.e., there are at least two VNFs mapped to NFV node n, and $\mathrm{w}_n(k) = 0$ otherwise. The loading factor of NFV node n during interval k, denoted by $\eta_n(k)$, consists of two parts, with a maximum value equal $\eta(k)$, which is upper bounded by a predefined constant η_U $(0 < \eta_U \leq 1)$, given by

$$\sum_{(r,h)\in\mathcal{V}} \left[\frac{P_n^{rh} b_r S_n^{rh}(k)}{\mathsf{C}_n} + \frac{\mathrm{w}_n(k)\mathrm{x}_n^{rh}(k)\mathtt{W}}{T_n} \right] \leq \eta(k) \leq \eta_U, \quad \forall n \in \mathcal{N} \tag{2.10}$$

where both useful time and switching time overhead of CPU resources in a polling period are taken into consideration. The left hand side of (2.10) is the expression for $\eta_n(k)$. The value of $\mathrm{w}_n(k)$ is confined by an inequality constraint of

$$\frac{\sum_{(r,h)\in\mathcal{V}} \mathrm{x}_n^{rh}(k) - 1}{|\mathcal{V}|} \leq \mathrm{w}_n(k) \leq \frac{\sum_{(r,h)\in\mathcal{V}} \mathrm{x}_n^{rh}(k)}{\mathsf{c}}, \quad \forall n \in \mathcal{N} \tag{2.11}$$

where c is an arbitrary value from $(1, 2]$.

Let $\boldsymbol{d}(k) = \{d_n^{rh}(k)\}$ be a positive continuous decision variable set for interval k, with $d_n^{rh}(k)$ denoting the average (dummy) delay on the queue associated with VNF $V_h^{(r)}$ at NFV node n. With Poisson traffic arrival and exponential packet processing time, the processing system is an M/M/1 queue. Then, $d_n^{rh}(k)$ is given by

$$d_n^{rh}(k) = \frac{1}{S_n^{rh}(k) - \mathrm{x}_n^{rh}(k)\lambda^{(r)}(k) + \epsilon_1}, \quad \forall (r,h) \in \mathcal{V},\ \forall n \in \mathcal{N} \tag{2.12}$$

where $d_n^{rh}(k)$ is a dummy delay only if $\mathrm{x}_n^{rh}(k) = 0$. There is an upper bound constraint for $d_n^{rh}(k)$, explicitly showing its relationship with $\mathrm{x}_n^{rh}(k)$:

$$0 < d_n^{rh}(k) \leq \mathrm{x}_n^{rh}(k)\mathsf{D}_r + \left[1 - \mathrm{x}_n^{rh}(k)\right] \frac{1}{\epsilon_1}, \quad \forall (r,h) \in \mathcal{V},\ \forall n \in \mathcal{N}. \tag{2.13}$$

For QoS satisfaction, the average E2E delay of SFC $r \in \mathcal{R}$ should not exceed upper bound D_r, represented as

$$\sum_{h \in \mathcal{H}_r} \sum_{n \in \mathcal{N}} \mathrm{x}_n^{rh}(k) d_n^{rh}(k) \leq \mathsf{D}_r, \quad \forall r \in \mathcal{R}. \tag{2.14}$$

To decide whether a subflow should be mapped to an extra virtual link, we consider two cases. In the first case, we have

$$y_{nn'}^{rh}(k) = 1 - \sum_{e \in \mathcal{E}} I_n^{e,1} I_{n'}^{e,2} \mathrm{x}_n^{rh}(k) \mathrm{x}_{n'}^{r(h+1)}(k), \quad \forall r \in \mathcal{R}, \forall h \in \{0\} \cup \mathcal{H}_r, \forall n \in \mathcal{N} \cup \mathcal{A}, \\ \forall n' \in \mathcal{N} \cup \mathcal{A} \backslash \{n\} \tag{2.15}$$

to ensure that $y_{nn'}^{rh}(k)$ equal 0 if (dummy) VNF $V_h^{(r)}$ and (dummy) VNF $V_{h+1}^{(r)}$ are mapped to node $n \in \mathcal{N} \cup \mathcal{A}$ and node $n' \in \mathcal{N} \cup \mathcal{A} \backslash \{n\}$ between which a virtual link exists. In the second case, we have

$$y_{nn}^{rh}(k) = 1 - \mathrm{x}_n^{rh}(k) \mathrm{x}_n^{r(h+1)}(k), \forall r \in \mathcal{R}, \forall h \in \{0\} \cup \mathcal{H}_r, \quad \forall n \in \mathcal{N} \cup \mathcal{A} \tag{2.16}$$

to ensure that $y_{nn}^{rh}(k)$ equal 0 if (dummy) VNF $V_h^{(r)}$ and (dummy) VNF $V_{h+1}^{(r)}$ are mapped to the same node $n \in \mathcal{N} \cup \mathcal{A}$.

In summary, the optimization problem is

$$\min_{\substack{\eta(k), \boldsymbol{B}(k), \boldsymbol{y}(k), \mathbf{x}(k), \boldsymbol{g}(k), \\ \boldsymbol{u}(k), \boldsymbol{S}(k), \mathbf{w}(k), \boldsymbol{d}(k)}} \quad O(k) \tag{2.17a}$$

$$\text{s.t.} \qquad (2.2) - (2.16) \tag{2.17b}$$

$$\boldsymbol{d}(k), \boldsymbol{u}(k) > 0, \tag{2.17c}$$

$$\mathbf{x}(k), \mathbf{w}(k), \boldsymbol{g}(k), \boldsymbol{y}(k) \in \{0, 1\}. \tag{2.17d}$$

Remark 1. *Problem* (2.17) *is non-convex due to constraints* (2.7), (2.10), (2.12), (2.14), (2.15), *and* (2.16).

2.3 Optimal MIQCP Solution

In problem (2.17), quadratic constraints (2.10), (2.14) (2.15), and (2.16) can be transformed to equivalent linear forms using the big-$\mathbb{M}$ method. Quadratic constraints (2.7) and (2.12)

cannot be linearized due to product terms of two continuous variables, but they can be replaced by combinations of linear constraints and rotated quadratic cone constraints. The new problem with transformed and replaced constraints is an MIQCP problem, which is not equivalent to the original problem. In this section, we discuss the relationship between the two problems.

By introducing an auxiliary nonnegative continuous decision variable set, $\boldsymbol{\zeta}(k) = \{\zeta_n(k)\}$, we linearize constraint (2.10) based on the big-$\mathbb{M}$ method with $\mathbb{M} = |\mathcal{V}|$, given by

$$\sum_{(r,h)\in\mathcal{V}} \frac{P_n^{rh} b_r S_n^{rh}(k)}{\mathsf{C}_n} T_n + \zeta_n(k)\mathtt{W} \leq \eta(k) T_n \leq \eta_U T_n, \quad \forall n \in \mathcal{N} \tag{2.18a}$$

$$\sum_{(r,h)\in\mathcal{V}} \mathrm{x}_n^{rh}(k) - |\mathcal{V}|\,[1 - \mathrm{w}_n(k)] \leq \zeta_n(k) \leq \sum_{(r,h)\in\mathcal{V}} \mathrm{x}_n^{rh}(k), \quad \forall n \in \mathcal{N} \tag{2.18b}$$

$$0 \leq \zeta_n(k) \leq |\mathcal{V}| \mathrm{w}_n(k), \quad \forall n \in \mathcal{N}. \tag{2.18c}$$

By introducing an auxiliary nonnegative continuous decision variable set, $\boldsymbol{\iota}(k) = \{\iota_n^{rh}(k)\}$, we linearize constraint (2.14) based on the big-$\mathbb{M}$ method with $\mathbb{M} = \frac{1}{\epsilon_1}$ as

$$\sum_{h\in\mathcal{H}_r} \sum_{n\in\mathcal{N}} \iota_n^{rh}(k) \leq \mathsf{D}_r, \quad \forall r \in \mathcal{R} \tag{2.19a}$$

$$d_n^{rh}(k) - \frac{1}{\epsilon_1}\left[1 - \mathrm{x}_n^{rh}(k)\right] \leq \iota_n^{rh}(k) \leq d_n^{rh}(k), \quad \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N} \tag{2.19b}$$

$$0 \leq \iota_n^{rh}(k) \leq \frac{1}{\epsilon_1} \mathrm{x}_n^{rh}(k), \quad \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N}. \tag{2.19c}$$

By introducing an auxiliary binary decision variable set, $\boldsymbol{\xi}(k) = \{\xi_{nn'}^{rh}(k)\}$, we get an equivalent linear form of constraint (2.15) and constraint (2.16) for $\forall r \in \mathcal{R}, \forall h \in \{0\} \cup \mathcal{H}_r$, and $\forall n \in \mathcal{N} \cup \mathcal{A}$, given by

$$\xi_{nn'}^{rh}(k) \leq \mathrm{x}_n^{rh}(k), \qquad \forall n' \in \mathcal{N} \cup \mathcal{A} \tag{2.20a}$$

$$\xi_{nn'}^{rh}(k) \leq \mathrm{x}_{n'}^{r(h+1)}(k), \qquad \forall n' \in \mathcal{N} \cup \mathcal{A} \tag{2.20b}$$

$$\xi_{nn'}^{rh}(k) \geq \mathrm{x}_n^{rh}(k) + \mathrm{x}_{n'}^{r(h+1)}(k) - 1, \quad \forall n' \in \mathcal{N} \cup \mathcal{A} \tag{2.20c}$$

$$y_{nn'}^{rh}(k) = 1 - \sum_{e\in\mathcal{E}} I_n^{e,1} I_{n'}^{e,2} \xi_{nn'}^{rh}(k), \quad \forall n' \in \mathcal{N} \cup \mathcal{A} \backslash \{n\} \tag{2.20d}$$

$$y_{nn}^{rh}(k) = 1 - \xi_{nn}^{rh}(k). \tag{2.20e}$$

Proposition 1. *With linearized constraints* (2.18), (2.19) *and* (2.20), *problem* (2.17) *can be transformed to an MIQCP problem, if constraint* (2.7) *is replaced by*

$$\varrho_{n,n'}^{rh}(k) = B_{n,n'}^{rh}(k) + \epsilon_1, \qquad \forall (r,h) \in \mathcal{V},\ n, n' \in \mathcal{N} \tag{2.21a}$$

$$\varrho_{n,n'}^{rh}(k) \geq \epsilon_1, \qquad \forall (r,h) \in \mathcal{V},\ n, n' \in \mathcal{N} \tag{2.21b}$$

$$u_{n,n'}^{rh}(k)\varrho_{n,n'}^{rh}(k) \geq \varsigma_h^{(r)}(k)^2, \quad \forall (r,h) \in \mathcal{V},\ n, n' \in \mathcal{N} \tag{2.21c}$$

$$\varsigma_h^{(r)}(k) = \sqrt{U_h^{(r)}(k)}, \qquad \forall (r,h) \in \mathcal{V} \tag{2.21d}$$

where $\boldsymbol{\varrho}(k) = \{\varrho_{n,n'}^{rh}(k)\}$ *and* $\boldsymbol{\varsigma}(k) = \{\varsigma_h^{(r)}(k)\}$ *are auxiliary continuous decision variable sets, and if constraint* (2.12) *is replaced by*

$$\varpi_n^{rh}(k) = S_n^{rh}(k) - \mathrm{x}_n^{rh}(k)\lambda^{(r)}(k) + \epsilon_1,\ \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N} \tag{2.22a}$$

$$\varpi_n^{rh}(k) \geq \epsilon_1, \qquad \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N} \tag{2.22b}$$

$$d_n^{rh}(k)\varpi_n^{rh}(k) \geq \Xi^2, \quad \forall (r,h) \in \mathcal{V}, \forall n \in \mathcal{N} \tag{2.22c}$$

$$\Xi = 1 \tag{2.22d}$$

where $\boldsymbol{\varpi}(k) = \{\varpi_n^{rh}(k)\}$ *is an auxiliary continuous decision variable set and* Ξ *is an auxiliary continuous decision variable. The optimality gap between the two problems is zero, i.e., an optimum of problem* (2.17) *is either a unique optimum or one of multiple optimal solutions to the MIQCP problem. Given an MIQCP optimum ("*$\star$*"), an optimum of problem* (2.17) *("*$*$*") can be obtained by Algorithm 1.*

Proof. The fundamental difference between the MIQCP problem and problem (2.17) lies in "$\geq$" signs in rotated quadratic cone constraints (2.21c) and (2.22c). If both constraints are active in an MIQCP optimum, i.e., all the "$\geq$" signs achieve equality, the MIQCP optimum is also an optimum of problem (2.17). Next, we discuss how the "$\geq$" signs affect the optimum.

First, assume that there is an inactive constraint (2.21c) in an MIQCP optimum, i.e., ${u_{n,n'}^{rh}}^{\star}(k)\left[{B_{n,n'}^{rh}}^{\star}(k) + \epsilon_1\right] > U_h^{(r)}(k)$. If ${B_{n,n'}^{rh}}^{\star}(k) = 0$, it does not affect the objective value. Thus, we consider only the case with ${B_{n,n'}^{rh}}^{\star}(k) > 0$. If ${B_{n,n'}^{rh}}^{\star}(k)$ is replaced by ${B_{n,n'}^{rh}}^{\circ}(k)$, with ${B_{n,n'}^{rh}}^{\circ}(k) < {B_{n,n'}^{rh}}^{\star}(k)$ and ${u_{n,n'}^{rh}}^{\star}(k)\left[{B_{n,n'}^{rh}}^{\circ}(k) + \epsilon_1\right] = U_h^{(r)}(k)$, all constraints are still satisfied. The objective value is unchanged if $\omega_2 = 0$, in which case $[{u_{n,n'}^{rh}}^{\star}(k),\ {B_{n,n'}^{rh}}^{\circ}(k)]$ is an optimal pair in another MIQCP optimum. Otherwise ($\omega_2 > 0$), the objective value can be further reduced, inferring that the assumption must be false.

Algorithm 1: Post-processing to MIQCP optimum

Input: $\eta^\star$, $\boldsymbol{B}^\star$, $\boldsymbol{y}^\star$, $\mathbf{x}^\star$, $\boldsymbol{g}^\star$, $\boldsymbol{u}^\star$, $\boldsymbol{S}^\star$, $\mathbf{w}^\star$, $\boldsymbol{d}^\star$.
Initialization $(* = \star)$.
for $(r,h) \in \mathcal{V}$, $n, n' \in \mathcal{N}$ **do**
  if $u^{rh}_{n,n'}{}^\star(k)\, \varrho^{rh}_{n,n'}{}^\star(k) > (\varsigma^{(r)}_h(k)^\star)^2$ *and* $B^{rh}_{n,n'}{}^\star(k) > 0$, **then**
    $B^{rh}_{n,n'}{}^*(k) = \frac{U^{(r)}_h(k)}{u^{rh}_{n,n'}{}^\star(k)} - \epsilon_1$
end
for $(r,h) \in \mathcal{V}$, $n \in \mathcal{N}$ **do**
  if $d^{rh\star}_n(k)\, \varpi^{rh\star}_n(k) > (c^\star)^2$ *and* $\mathrm{x}^{rh\star}_n(k) == 1$, **then**
    if $\omega_1 > 0$, **then**
      $d^{rh*}_n(k) = \frac{1}{S^{rh\star}_n(k) - \mathrm{x}^{rh\star}_n(k)\,\lambda^{(r)}(k) + \epsilon_1}$
    end
    if $\omega_1 == 0$, **then**
      $S^{rh*}_n(k) = \frac{1}{d^{rh\star}_n(k)} + \mathrm{x}^{rh\star}_n(k)\,\lambda^{(r)}(k) - \epsilon_1$
    end
  end
end
if $\omega_1 == 0$, **then** *calculate* $\eta^*(k)$
Output: η^*, $\boldsymbol{B}^*$, $\boldsymbol{y}^*$, $\mathbf{x}^*$, $\boldsymbol{g}^*$, $\boldsymbol{u}^*$, $\boldsymbol{S}^*$, $\mathbf{w}^*$, $\boldsymbol{d}^*$.

Second, assume that there is an inactive constraint (2.22c) in an MIQCP optimum, then $d^{rh\star}_n(k)\left[S^{rh\star}_n(k) - \mathrm{x}^{rh\star}_n(k)\lambda^{(r)}(k) + \epsilon_1\right] > 1$. Similarly, we consider only the constraint with $\mathrm{x}^{rh\star}_n(k) = 1$. There are four cases, depending on ω_1 and $\eta_n(k)$.

Case 1: $\omega_1 > 0$, and NFV node n is the only one with a loading factor of $\eta^\star(k)$ (i.e., dominating NFV node). If $S^{rh\star}_n(k)$ is replaced by $S^{rh\circ}_n(k)$, with $d^{rh\star}_n(k)[S^{rh\circ}_n(k) - \mathrm{x}^{rh\star}_n(k)\lambda^{(r)}(k) + \epsilon_1] = 1$, all constraints are satisfied but $\eta^\star(k)$ can be further reduced. Hence, the assumption must be false;

Case 2: $\omega_1 > 0$ and $\eta_n(k) < \eta^\star(k)$ (i.e., non-dominating NFV node). If $d^{rh\star}_n(k)$ is replaced by $d^{rh\circ}_n(k)$, with $d^{rh\circ}_n(k)[S^{rh\star}_n(k) - \mathrm{x}^{rh\star}_n(k)\lambda^{(r)}(k) + \epsilon_1] = 1$, all constraints are satisfied with the objective value unchanged. Thus, $[d^{rh\circ}_n(k),\, S^{rh\star}_n(k)]$ is an optimal pair in another MIQCP optimum;

Case 3: $\omega_1 > 0$, and there is more than one NFV node including NFV node n with loading factor $\eta^\star(k)$. There must be at least one of them satisfying an active constraint (2.22c). One such NFV node is selected as the dominating NFV node, and others are seen as non-dominating NFV nodes;

Case 4: $\omega_1 = 0$, and $\eta(k)$ is not optimized. If we replace $S_n^{rh^\star}(k)$ by $S_n^{rh^\circ}(k)$, all constraints are satisfied and the objective value is unchanged. Thus, $[d_n^{rh^\star}(k), S_n^{rh^\circ}(k)]$ is an optimal pair in another MIQCP optimum.

In summary, an MIQCP optimum with inactive constraints in (2.21c) and (2.22c) can be mapped to another MIQCP optimum with active constraints in (2.21c) and (2.22c), without affecting other constraints and the objective value. The mapped MIQCP optimum is also the optimum of problem (2.17). The mapping algorithm is provided in Algorithm 1.

Remark 2. *The MIQCP problem is NP-hard.*

Proof. To prove the NP-hardness, it is sufficient to consider a special case in which services with $\mathsf{D}_r \to \infty$ are embedded in a fully-connected virtual resource pool. We also consider zero VNF state size, zero switching time overhead, and sufficiently large processing resource capacity for each NFV node holding all VNFs without overloading [29,31]. In such a case, the MIQCP problem can be reduced from a multiprocessor scheduling problem [75]. The multiprocessor scheduling problem minimizes the maximum load among a number of processors which are assigned with a number of tasks with different loads, which is proved to be NP-hard.

2.4 Heuristic Solution

Although problem (2.17) can be solved by the optimal MIQCP solution according to Proposition 1, the computational time is high due to NP-hardness of the MIQCP problem. In this section, we propose a low-complexity modular heuristic solution to problem (2.17). We consider only the case where all components in objective function (2.1) are jointly optimized, i.e., $\omega_1, \omega_2, \omega_3 > 0$. In this case, we assume that one VNF migration is penalized more than imbalanced loading (i.e., $\eta(k)$ reaching its upper bound η_U). Then, the condition of $\omega_1 \eta_U < \omega_2$ should be satisfied in the worst case, if all VNF migrations incur the same transmission resource overhead for state transfer and require no extra virtual links

for flow rerouting. Accordingly, in the proposed algorithm, we first minimize the number of overloaded NFV nodes with loading factors greater than η_U, and make migration decisions at overloaded NFV nodes, after which $\eta(k)$ is equal to η_U. Afterwards, $\eta(k)$ is further reduced for load balancing. The algorithm is insensitive to ω_1 but sensitive to $\frac{\omega_2}{\omega_3}$, due to reconfiguration overhead aware migration decisions. Therefore, it provides a sub-optimal solution to problem (17) with $\omega_1\eta_U < \omega_2$.

2.4.1 Overview

The heuristic algorithm is to determine a migration and resource allocation plan for interval k in the presence of predicted traffic variations (i.e., from $\{\lambda^{(r)}(k-1)\}$ to $\{\lambda^{(r)}(k)\}$). We first find if and where NFV node resource overloading would happen due to traffic variations, based on three factors. The first is node mapping, denoted by $\{\mathrm{x}_n^{rh}(k)\}$; the second is hop (VNF) delay bounds, denoted by $\{\mathsf{D}^{rh}(k)\}$, with $\sum_{h\in\mathcal{H}_r}\mathsf{D}^{rh}(k) = \mathsf{D}_r$; the third is NFV node loading factor threshold, denoted by η_{th}. With current node mapping and hop delay bounds, i.e., $\mathrm{x}_n^{rh}(k) = \mathrm{x}_n^{rh}(k-1)$ and $\mathsf{D}^{rh}(k) = \sum_{n\in\mathcal{N}}\mathrm{x}_n^{rh}(k-1)d_n^{rh}(k-1)$, we calculate NFV node loading factors with traffic rates, $\{\lambda^{(r)}(k)\}$, based on the M/M/1 queueing model. By comparing the NFV node loading factors with threshold η_{th} (initial value set as η_U), a set of overloaded NFV nodes is identified as potential bottlenecks.

Reconfiguration overhead reduction

Even if potential bottlenecks are identified, it is possible that migration is not necessary. For a given η_{th} value, how an E2E delay requirement is decomposed into hop delay bounds makes a difference on the number of overloaded NFV nodes. By making hop delay bounds less stringent on overloaded NFV nodes and more stringent on underloaded NFV nodes, it is possible to reduce the number of overloaded NFV nodes. The basic idea is as follows. If an SFC traverses both overloaded and underloaded NFV nodes, loading factors of the underloaded ones are increased to η_{th}, by shrinking corresponding hop delay bounds, and loading factors of the overloaded ones are decreased, by enlarging corresponding hop delay bounds. The strategy is referred to as delay scaling. Delay scaling is performed iteratively, until there is no SFC traversing both overloaded and underloaded NFV nodes. The iterative delay scaling procedure with given threshold, η_{th}, is referred to as redistribution of hop delay bounds.

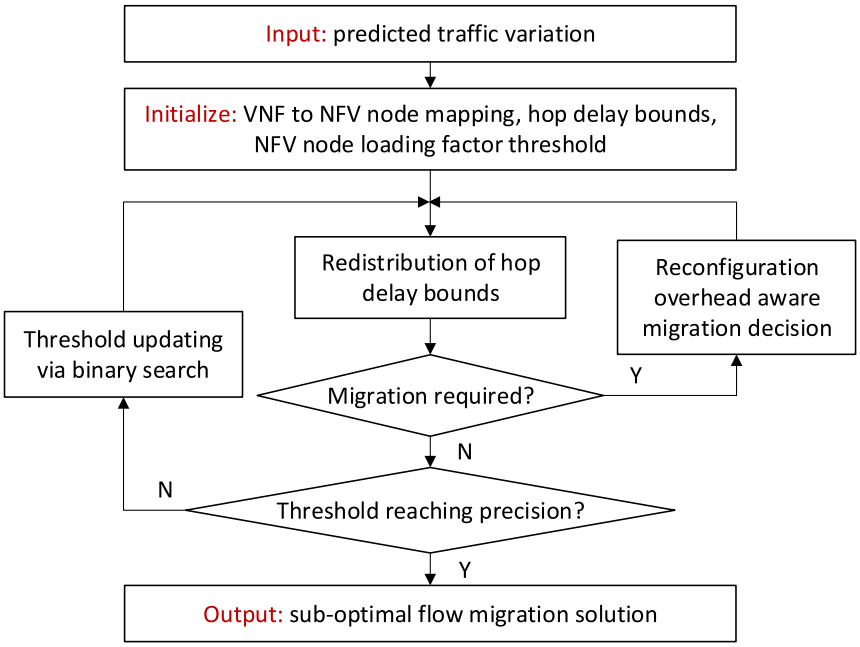


Figure 2.4: Flowchart of the heuristic algorithm for dynamic flow migration.

If the number of overloaded NFV nodes is reduced to zero after redistribution of hop delay bounds, no migration is required. Otherwise, migration is necessary to overcome resource overloading. Migration decisions are made sequentially, i.e., only a pair of variables in set $\{\mathrm{x}_n^{rh}(k)\}$ is updated in one migration decision, each followed by a redistribution of hop delay bounds, until no more migration is required.

With alternate migration decision and redistribution of hop delay bounds, reconfiguration overhead is greedily reduced in two ways. One is the potential reduction of overloaded NFV nodes by redistribution of hop delay bounds. The other is consideration of reconfiguration overhead in migration decision.

Load balancing

If no potential bottlenecks are detected or all detected potential bottlenecks are removed by migration and redistribution of hop delay bounds, load balancing is the only remaining objective. NFV node loading factors are gradually balanced by iterative redistribution of hop delay bounds with threshold updating. The threshold, η_{th}, is updated from binary

search, until it reaches sufficient precision.

More details on redistribution of hop delay bounds are given in Subsection 2.4.2, with pseudo code presented in Algorithm 2. Migration decision is discussed in Subsection 2.4.3, and threshold updating is discussed in Subsection 2.4.4. Finally, the heuristic algorithm is presented in Algorithm 3, with a flowchart given in Fig. 2.4.

2.4.2 Redistribution of Hop Delay Bounds

Classification. With given threshold η_{th} and a given set of $\{\mathsf{D}^{rh}(k)\}$, the loading factor of NFV node $n \in \mathcal{N}$ in the presence of traffic variations is calculated as

$$\eta_n(k) = \sum_{(r,h)\in\mathcal{V}} \left(\frac{P_n^{rh} b_r S_n^{rh}(k)}{\mathsf{C}_n} + \frac{\mathrm{w}_n(k)\mathrm{x}_n^{rh}(k)\mathtt{W}}{T_n} \right) \tag{2.23}$$

where $\mathrm{w}_n(k)$ is calculated from (2.11) and $S_n^{rh}(k)$ is given by

$$S_n^{rh}(k) = \left(\lambda^{(r)}(k) + \frac{1}{\mathsf{D}^{rh}(k)} \right) \mathrm{x}_n^{rh}(k). \tag{2.24}$$

Three sets of NFV nodes are identified: $\mathcal{N}_O = \{n \in \mathcal{N} \,|\, \eta_n(k) > \eta_{th}\}$ consisting of overloaded NFV nodes, $\mathcal{N}_U = \{n \in \mathcal{N} \,|\, \eta_n(k) < \eta_{th}\}$ for underloaded NFV nodes, and $\mathcal{N}_E = \{n \in \mathcal{N} \,|\, \eta_n(k) = \eta_{th}\}$. Let binary variable $X_n^{(r)}(k)$ indicate whether SFC r traverses NFV node n during interval k, with $X_n^{(r)}(k) = 1$ if $\sum_{h\in\mathcal{H}_r} \mathrm{x}_n^{rh}(k) > 0$, and $X_n^{(r)}(k) = 0$ otherwise. Let $\mathtt{f}_1^{(r)}$ be a binary flag indicating whether SFC r traverses any overloaded NFV nodes, with $\mathtt{f}_1^{(r)} = 1$ if $\sum_{n\in\mathcal{N}_O} X_n^{(r)}(k) > 0$, and $\mathtt{f}_1^{(r)} = 0$ otherwise. Set $\mathcal{N}_U$ is divided into two subsets, i.e., $\mathcal{N}_U = \mathcal{N}_{U,U} \cup \mathcal{N}_{U,O}$, where $\mathcal{N}_{U,U} = \{n \in \mathcal{N}_U \,|\, \sum_{r\in\mathcal{R}} X_n^{(r)}(k)\mathtt{f}_1^{(r)} = 0\}$ is a set of underloaded NFV nodes on which no SFCs traverse other overloaded NFV nodes, and $\mathcal{N}_{U,O} = \{n \in \mathcal{N}_U \,|\, \sum_{r\in\mathcal{R}} X_n^{(r)}(k)\mathtt{f}_1^{(r)} > 0\}$ is a set of underloaded NFV nodes on which at least one SFC traverses other overloaded NFV nodes. Let $\mathtt{f}_2^{(r)}$ be a binary flag indicating whether SFC r traverses any NFV nodes in $\mathcal{N}_{U,O}$, with $\mathtt{f}_2^{(r)} = 1$ if $\sum_{n\in\mathcal{N}_{U,O}} X_n^{(r)}(k) > 0$, and $\mathtt{f}_2^{(r)} = 0$ otherwise. Accordingly, SFCs are classified into four categories: SFC category I with $\mathtt{f}_1^{(r)} = 1$ and $\mathtt{f}_2^{(r)} = 0$, SFC category II with $\mathtt{f}_1^{(r)} = \mathtt{f}_2^{(r)} = 1$, SFC category III with $\mathtt{f}_1^{(r)} = 0$ and $\mathtt{f}_2^{(r)} = 1$, and SFC category IV with $\mathtt{f}_1^{(r)} = \mathtt{f}_2^{(r)} = 0$, as shown in Fig. 2.5.

Update and iteration. Define two sets of delay scaling factors, vertical delay scaling factors $\{\tilde{\alpha}_n(k)\}$ and horizontal delay scaling factors $\{\tilde{\beta}^{(r)}(k)\}$, with initial values of 1. A two-step delay scaling strategy is proposed as follows.

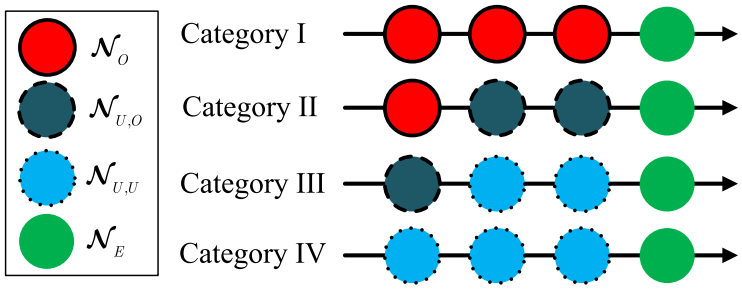


Figure 2.5: Four SFC categories based on NFV node loading factors.

Step I - Delay scaling for SFC category III

Hop delay bounds for SFC category III are relaxed on NFV nodes in $\mathcal{N}_{U,O}$, to release resources for SFC category II, by making hop delay bounds more stringent on NFV nodes in $\mathcal{N}_{U,U}$.

First, the loading factor of NFV node $n \in \mathcal{N}_{U,U}$ is increased to η_{th}, by shrinking hop delay bounds for SFC category III on NFV node n by a positive factor, $\tilde{\alpha}_n(k)$, less than 1, as derived in Appendix A and given by

$$\tilde{\alpha}_n(k) = \frac{\sum_{(r,h)\in\mathcal{V}} \frac{P_n^{rh} b_r}{\mathsf{D}^{rh}(k)} \mathrm{x}_n^{rh}(k) \mathtt{f}_2^{(r)}}{[\eta_{th} - \eta_n(k)]\, \mathsf{C}_n + \sum_{(r,h)\in\mathcal{V}} \frac{P_n^{rh} b_r}{\mathsf{D}^{rh}(k)} \mathrm{x}_n^{rh}(k) \mathtt{f}_2^{(r)}}. \tag{2.25}$$

The preceding delay scaling, called vertical delay scaling, is applied to multiple SFCs belonging to category III on NFV node $n \in \mathcal{N}_{U,U}$. Then, hop delay bounds for SFC r in category III on NFV nodes in $\mathcal{N}_{U,O}$ are relaxed by a factor, $\tilde{\beta}^{(r)}(k)$, larger than 1, given by

$$\tilde{\beta}^{(r)}(k) = \frac{\mathsf{D}_r - \sum_{h\in\mathcal{H}_r} \left(\mathsf{D}^{rh}(k) \sum_{n\in\mathcal{N}_{U,U}\cup\mathcal{N}_E} \mathrm{x}_n^{rh}(k) \right)}{\mathsf{D}_r - \sum_{h\in\mathcal{H}_r} \left(\mathsf{D}_p^{rh}(k) \sum_{n\in\mathcal{N}_{U,U}\cup\mathcal{N}_E} \mathrm{x}_n^{rh}(k) \right)} \tag{2.26}$$

where $\mathsf{D}_p^{rh}(k)$ is the old value of $\mathsf{D}^{rh}(k)$ before vertical delay scaling. The preceding delay scaling, called horizontal delay scaling, is applied to multiple hops in an SFC affected by vertical delay scaling. Based on (2.23), $\{\eta_n(k)\}$ is updated.

Algorithm 2: Redistribution of hop delay bounds

1 **Input:** η_{th}, $\{\mathrm{x}_n^{rh}(k)\}$, $\{\mathsf{D}^{rh}(k)\}$
2 Calculate $\{\eta_n(k)\}$, $\mathcal{N}_O$, $\mathcal{N}_{U,U}$, $\mathcal{N}_{U,O}$, $\mathcal{N}_E$, $\{\mathtt{f}_1^{(r)}\}$, $\{\mathtt{f}_2^{(r)}\}$.
3 **while** $\mathcal{N}_{U,O} \neq \varnothing$ *and* $\sum_{r\in\mathcal{R}} \mathtt{f}_1^{(r)} > 0$ **do**
4 $\{\tilde{\alpha}_n(k)\} = 1$; $\{\tilde{\beta}^{(r)}(k)\} = 1$.
5 **if** *in the first while loop,* **then**
6 Vertical delay scaling at NFV nodes in $\mathcal{N}_{U,U}$.
7 Horizontal delay scaling for SFC category III at NFV nodes in $\mathcal{N}_{U,O}$.
8 Update $\{\eta_n(k)\}$.
9 **end**
10 Vertical delay scaling at NFV nodes in $\mathcal{N}_{U,O}$.
11 Horizontal delay scaling for SFC category II at NFV nodes in $\mathcal{N}_O$.
12 Update $\{\eta_n(k)\}$, $\mathcal{N}_O$, $\mathcal{N}_{U,U}$, $\mathcal{N}_{U,O}$, $\mathcal{N}_E$, $\{\mathtt{f}_1^{(r)}\}$, $\{\mathtt{f}_2^{(r)}\}$.
13 **end**
14 **Output:** $\{\mathsf{D}^{rh}(k)\}$, $\{\eta_n(k)\}$, $\mathcal{N}_O$, $\mathcal{N}_{O,1}$, $\{\mathtt{f}_1^{(r)}\}$.

Step II - Delay scaling for SFC category II

More resources available at NFV nodes in $\mathcal{N}_{U,O}$ from Step I are allocated to SFC category II. First, vertical delay scaling is applied to NFV nodes in $\mathcal{N}_{U,O}$ to increase their loading factors to η_{th}, through scaling hop delay bounds for SFC category II on it by a vertical scaling factor. Then, horizontal delay scaling is applied to SFC category II, by relaxing hop delay bounds for SFC r in category II on NFV nodes in $\mathcal{N}_O$ by a horizontal scaling factor. The scaling factors are similar to that in Step I, and details are omitted.

After the delay scaling procedures, NFV node loading factors, NFV node classification and SFC categories are updated. If $\mathcal{N}_{U,O} \neq \varnothing$ and $\sum_{r\in\mathcal{R}} \mathtt{f}_1^{(r)} > 0$, i.e., there is at least one SFC in category II, it is possible to further reduce the number of overloaded NFV nodes through Step II, so Step II is performed iteratively until the condition is violated. The outputs of Algorithm 2 are shown in line 14, where $\mathcal{N}_{O,1} = \{n \in \mathcal{N}_O | \sum_{r\in\mathcal{R}} X_n^{(r)}(k) = 1\}$ denotes a set of overloaded NFV nodes traversed by a single SFC.

2.4.3 Migration Decision

When one migration is required, a migration decision procedure is to select one bottleneck NFV node, one SFC to migrate, and one target NFV node. Migration decisions are made greedily to reduce the reconfiguration overhead. First, a candidate bottleneck NFV node set, $\mathcal{B}$, with $|\mathcal{B}| = |\mathcal{R}|$, is determined. For an SFC, the traversed NFV node with largest hop delay bound is selected as a candidate bottleneck. Then, bottleneck NFV node, n_b, is determined in three cases. In the first case with $\mathcal{B} \cap \mathcal{N}_O \neq \varnothing$, an NFV node in $\mathcal{B} \cap \mathcal{N}_O$ with the largest number of SFCs is selected, given by

$$n_b = \underset{n \in \mathcal{B} \cap \mathcal{N}_O}{\operatorname{argmax}} \sum_{r \in \mathcal{R}} X_n^{(r)}(k). \tag{2.27}$$

In the second case with $\mathcal{B} \cap \mathcal{N}_O = \varnothing$ and $\mathcal{N}_O \backslash \mathcal{N}_{O,1} \neq \varnothing$, n_b is an NFV node in $\mathcal{N}_O \backslash \mathcal{N}_{O,1}$ with the largest loading factor,

$$n_b = \underset{n \in \mathcal{N}_O \backslash \mathcal{N}_{O,1}}{\operatorname{argmax}} \eta_n(k). \tag{2.28}$$

In the third case with $\mathcal{B} \cap \mathcal{N}_O = \mathcal{N}_O \backslash \mathcal{N}_{O,1} = \varnothing$, an NFV node in $\mathcal{B}$ whose SFCs traverse the largest number of overloaded NFV nodes is selected, given by

$$n_b = \underset{n \in \mathcal{B}}{\operatorname{argmax}} \sum_{r \in \mathcal{R}} \left(X_n^{(r)}(k) \sum_{n' \in \mathcal{N}_O} X_{n'}^{(r)}(k) \right). \tag{2.29}$$

Next, an SFC to migrate from n_b and a target NFV node to accommodate the migrated SFC are jointly selected to minimize the reconfiguration overhead, i.e., the weighted sum of normalized transmission resource overhead for state transfer and number of extra virtual links for flow rerouting. In this way, ω_2 and ω_3 are considered in the heuristic algorithm. If there are multiple choices, an SFC with the largest resource demand is migrated to the closest target NFV node.

2.4.4 Coordination with Threshold Update

Let binary variable, `v`, indicate whether migration is required to overcome resource overloading. It is set as 0 initially and updated iteratively. Let η_Δ be a step size to update η_{th},

with initial value $\eta_{\Delta,0}$ and being updated before each η_{th} update. A precision, η_{Δ}^{ϵ}, for η_{Δ} is set as a stop condition.

After initialization in Algorithm 3 (lines 2-3), a redistribution of hop delay bounds is performed to check whether migration is required. Based on outputs of Algorithm 2, $\mathtt{v}$, η_{th} and η_{Δ} are updated in three cases, as shown in lines 8-12.

Update for $\mathtt{v}$ and η_{th}

In the first case, there are no remaining overloaded NFV nodes, i.e., $\sum_{r\in\mathcal{R}} \mathtt{f}_1^{(r)} = 0$. Then $\mathtt{v} = 0$, and η_{th} is reduced by a step. In the other two cases, there are still overloaded NFV nodes but $\mathcal{N}_{U,O} = \varnothing$, meaning that delay scaling is not sufficient to deal with resource overloading on NFV nodes, but either at least one migration or increasing η_{th} by a step is required, depending on the η_{th} value. In the second case with $\eta_{th} = \eta_U$, at least one migration is needed, i.e., $\mathtt{v} = 1$, and η_{th} should remain η_U to check whether more migrations are required after a migration decision is made. In the third case with $\eta_{th} < \eta_U$, no more migrations are required since η_{th} has been reduced by at least one step in previous updates, thus $\mathtt{v} = 0$ and η_{th} is increased by a step. With the updates for $\mathtt{v}$ and η_{th}, redistribution of hop delay bounds is performed iteratively until no more migrations are required and the precision of η_{Δ} reaches η_{Δ}^{ϵ}.

Update for step size η_{Δ}

Step size η_{Δ} plays a key role in guaranteeing algorithm convergence. For example, a constant η_{Δ} equal to η_{Δ}^{ϵ} guarantees precision but makes the algorithm slow to converge due to a potential oscillation of η_{th} around its optimal value. Therefore, we employ the following strategy to update η_{Δ}. If the outputs of Algorithm 2 fall into the second case where a migration is required, η_{Δ} remains a constant equal to $\eta_{\Delta,0}$. After all migrations are performed, the outputs of Algorithm 2 correspond to the first case, and η_{th} should be reduced by a constant step size η_{Δ} equal to $\eta_{\Delta,0}$. Until the first time that the outputs of Algorithm 2 fall into the third case, η_{Δ} starts to be reduced by half before each η_{th} update.

Algorithm 3: Heuristic algorithm for problem (2.17)

Input: Step size $\eta_{\Delta,0}$, precision η_{Δ}^{ϵ}
Initialize: $\{\mathrm{x}_n^{rh}(k)\}$, $\{\mathsf{D}^{rh}(k)\}$
Let: $\mathtt{v}=0$, $\eta_{th}=\eta_U$, $\eta_{\Delta}=\eta_{\Delta,0}$
while $\mathtt{v}==1$ *or* $\eta_{\Delta}>\eta_{\Delta}^{\epsilon}$ **do**
  if $\mathtt{v}==1$, **then**
    Update $\{\mathrm{x}_n^{rh}(k)\}$ according to the migration decision making procedure.
  end
  Update $\{\mathsf{D}^{rh}(k)\}$, $\{\eta_n(k)\}$, $\mathcal{N}_O$, $\mathcal{N}_{O,1}$, $\{\mathtt{f}_1^{(r)}\}$ according to Algorithm 2.
  if $\sum_{r\in\mathcal{R}}\mathtt{f}_1^{(r)}==0$, **then**
    if $\eta_{\Delta}\neq\eta_{\Delta,0}$, **then** $\eta_{\Delta}=\eta_{\Delta}/2$
    $\eta_{th}=\eta_{th}-\eta_{\Delta}$, $\mathtt{v}=0$
  end
  else if $\eta_{th}==\eta_U$, **then** $\mathtt{v}=1$
  else $\eta_{\Delta}=\eta_{\Delta}/2$, $\eta_{th}=\eta_{th}+\eta_{\Delta}$, $\mathtt{v}=0$
end
Output: Sub-optimal solution to problem (2.17).

2.4.5 Complexity Analysis

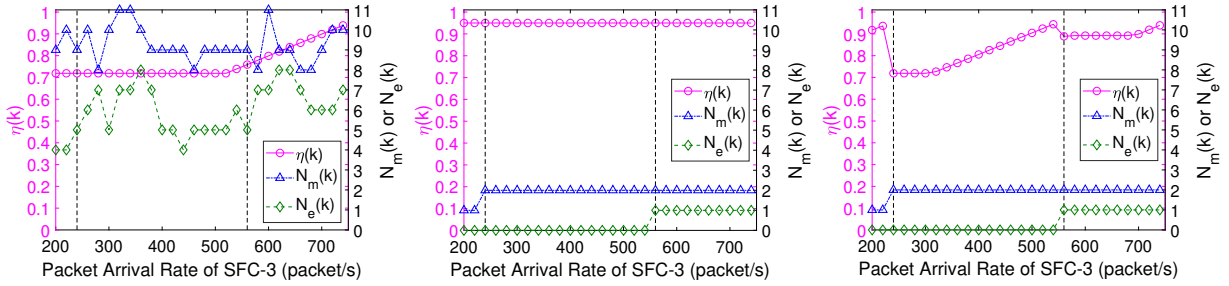
We first analyze the time complexity of Algorithm 2. Delay scaling for SFC category III is performed once, using at most $\mathcal{O}(\sum_{n\in\mathcal{N}}|\mathcal{V}|)$ time. Delay scaling for SFC category II is performed iteratively until there are no new NFV nodes in $\mathcal{N}_{U,O}$ or there are no overloaded SFCs. The worst case happens when each round of delay scaling for SFC category II transforms a single NFV node in $\mathcal{N}_O$ to a new NFV node in $\mathcal{N}_{U,O}$, consuming $\mathcal{O}(\sum_{n\in\mathcal{N}_O}\sum_{n\in\mathcal{N}}|\mathcal{V}|)$ time. Thus, the complexity of Algorithm 2 is $\mathcal{O}(\sum_{n\in\mathcal{N}_O}\sum_{n\in\mathcal{N}}|\mathcal{V}|)$, upper bounded by $\mathcal{O}(|\mathcal{N}|^2|\mathcal{V}|)$. The complexity of the migration decision procedure is dominated by the selection of SFC to migrate in the third case, which requires a running time of $\mathcal{O}(|\mathcal{N}||\mathcal{R}|^2)$. In Algorithm 3, at most $|\mathcal{V}|$ sequential migration decisions are performed followed by $[\frac{1}{\eta_{\Delta,0}}+log_2(\frac{\eta_{\Delta,0}}{\eta_{\Delta}^{\epsilon}})]$ iterations of threshold updating. In each iteration, hop delay bounds are readjusted. Therefore, the worst case running time of Algorithm 3 is $|\mathcal{V}|[\mathcal{O}(|\mathcal{N}||\mathcal{R}|^2)+\mathcal{O}(|\mathcal{N}|^2|\mathcal{V}|)]+[\frac{1}{\eta_{\Delta,0}}+log_2(\frac{\eta_{\Delta,0}}{\eta_{\Delta}^{\epsilon}})]\mathcal{O}(|\mathcal{N}|^2|\mathcal{V}|)$, which is simplified to $\mathcal{O}(|\mathcal{N}|^2|\mathcal{V}|^2)$ when $|\mathcal{R}|^2<|\mathcal{N}||\mathcal{V}|$.

2.5 Performance Evaluation

In this section, simulation results are presented to evaluate the MIQCP and heuristic solutions for the delay-aware flow migration problem. Two time intervals are considered: $(k-1)$ and k, representing the current and next time intervals respectively. We use two mesh networks with 64 NFV nodes and 256 NFV nodes to represent the virtual resource pool. Virtual links exist only between neighboring NFV nodes. In the 64-node network, we consider fixed SFC mapping for time interval $(k-1)$, with three SFCs initially mapped to the virtual resource pool. Specifically, SFC 3 shares two NFV nodes with SFC 1 and one of them also with SFC 2. In the 256-node network, we consider different numbers of SFCs, with [3, 5] VNFs in each one, randomly distributed in the network during time interval $(k-1)$. We set a ratio of 0.01 between the switching time and the CPU polling period. The upper bound, η_U, for $\eta(k)$, is 0.95. The average E2E delay requirement for each SFC is 0.02 s, and the maximal tolerable service downtime is 0.005 s. For VNF states, the size is a constant, equal to 10 *bytes*, thus requiring at least a transmission rate B_{min} of 16 *kbit/s* for a state transfer. Under the simulation setup, the total normalized transmission resource overhead for state transfer is equal to the total number of migrations. For the weights, we set $\omega_2 = 2\omega_3$. Then, $\omega_1 < \frac{2}{3\eta_U+2} = 0.4123$ should be satisfied to penalize migration more than imbalanced loading. In this case, 0.4123 is the worst-case boundary for ω_1, to guarantee the penalization preference if ω_1 is less than the boundary. We implement both the MIQCP and heuristic solutions in python. We use NetworkX to simulate the network scenario, and Gurobi python interface to solve the MIQCP problem.

2.5.1 Load Balancing and Reconfiguration Overhead Trade-off

We use the 64-node network with three SFCs to evaluate the performance of the MIQCP solution with varying traffic load under three sets of weights in (2.1), and investigate the trade-off between load balancing and reconfiguration overhead. For traffic load during interval k, we have $\lambda^{(1)}(k) = 600$ packet/s, $\lambda^{(2)}(k) = 200$ packet/s, and vary $\lambda^{(3)}(k)$ from 200 packet/s to 740 packet/s. Beyond 740 packet/s, the problem becomes infeasible due to processing resource constraints and average E2E delay constraints. Performance metrics are the maximum NFV node loading factor, $\eta(k)$, the number of migrations, $N_m(k)$, and the number of extra virtual links, $N_e(k)$, for flow rerouting. We explore three sets of weights. For $\{\omega_1,\ \omega_2,\ \omega_3\} = \{1,\ 0,\ 0\}$, the reconfiguration overhead is not optimized but load



(a) LBFM strategy (b) MOFM strategy (c) HFM strategy

Figure 2.6: Performance of three flow migration strategies with respect to $\lambda^{(3)}(k)$.

balancing is the focus, corresponding to a load balancing flow migration (LBFM) strategy. For $\{\omega_1, \omega_2, \omega_3\} = \{0, \frac{2}{3}, \frac{1}{3}\}$, $\eta(k)$ is not optimized but reconfiguration overhead reduction is emphasized, corresponding to a minimum overhead flow migration (MOFM) strategy. For $\{\omega_1, \omega_2, \omega_3\} = \{0.4, 0.4, 0.2\}$, both load balancing and reconfiguration overhead reduction are important, corresponding to a hybrid flow migration (HFM) strategy. Fig. 2.6 shows performance of three strategies with the increase of $\lambda^{(3)}(k)$, for $\eta_U = 0.95$.

LBFM strategy: It is observed that $\eta(k)$ is dominated by SFC 1 when $\lambda^{(3)}(k)$ is relatively small, showing a flat trend first, but turns to be dominated by SFC 3 with the increase of $\lambda^{(3)}(k)$. Both $N_m(k)$ and $N_e(k)$ are high and vary with the traffic load, since they are not optimized. SFCs separate from each other even at a relatively low traffic load to balance traffic loads in the virtual resource pool.

MOFM strategy: Both $N_m(k)$ and $N_e(k)$ show a step-wise increasing trend with the increase of $\lambda^{(3)}(k)$. However, $\eta(k)$ is fixed at η_U, since it is not optimized.

HFM strategy: A trade-off among performance metrics is observed. With the increase of $\lambda^{(3)}(k)$, $\eta(k)$ drops sharply when $N_m(k)$ or $N_e(k)$ is increased by 1. When $N_m(k)$ and $N_e(k)$ stay stable, $\eta(k)$ shows either a linear increasing or a flat trend. Compared with LBFM and MOFM strategies, HFM strategy approaches to the lower bounds of $N_m(k)$ and $N_e(k)$ determined by the MOFM strategy, while keeping $\eta(k)$ at a medium level.

2.5.2 Average End-to-End Delay Performance

We carry out packet-level simulations using network simulator OMNeT++ to evaluate average E2E delay of SFC 3 with and without flow migration, under the same network and

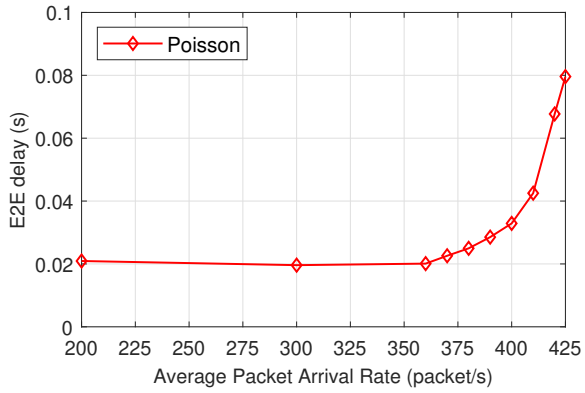


Figure 2.7: Average E2E delay without flow migration.

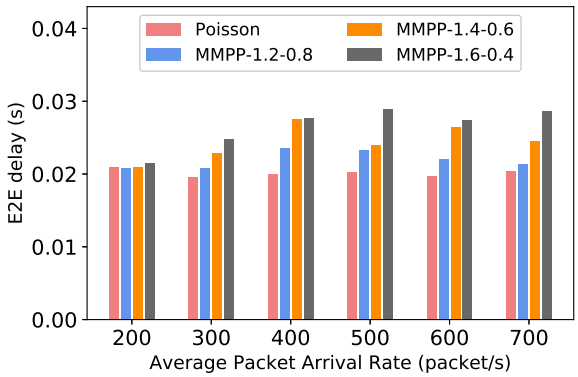


Figure 2.8: Average E2E delay comparison with flow migration.

SFC settings as in Subsection 2.5.1. For average traffic rates, we set $\lambda^{(1)}(k) = \lambda^{(2)}(k) = 200$ packet/s and increase $\lambda^{(3)}(k)$ from 200 packet/s. To verify the effectiveness and accuracy of our flow migration model in the presence of traffic burstiness, not only Poisson but also Markov-modulated Poisson process (MMPP) packet arrivals are simulated. For each traffic arrival pattern, we collect sufficient packet delay information to estimate the average E2E delay. We use a two-state MMPP model with same transition rate between states and an average rate of $\lambda^{(3)}(k)$. We use "MMPP-q_1-q_2" to represent the MMPP traffic model, where q_1 and q_2 are ratios between state-dependent rates and $\lambda^{(3)}(k)$, with $q_1 + q_2 = 2$. For example, for "MMPP-1.6-0.4" traffic model with $\lambda^{(3)}(k) = 500$ packet/s, the state-dependent rates are 800 packet/s and 200 packet/s respectively.

Fig. 2.7 shows the average E2E delay without flow migration, in which a flat trend is observed, followed by an exponential increasing trend, for Poisson traffic arrival with increasing rate from 200 packet/s to 425 packet/s. The flat trend corresponds to feasible traffic rates for E2E delay guarantee with local processing resource scaling. Beyond 360 packet/s, local resources are not sufficient, resulting in an exponential increase of E2E delay. Fig. 2.8 shows the average E2E delay with flow migration. We observe that the E2E delay requirement is satisfied for Poisson traffic with rate in [200, 700] packet/s, inferring that more traffic can be accommodated from services which originally share some NFV nodes on their E2E paths, with joint flow migration and processing resource scaling. At a certain average rate, the E2E delay performance degrades with more traffic burstiness.

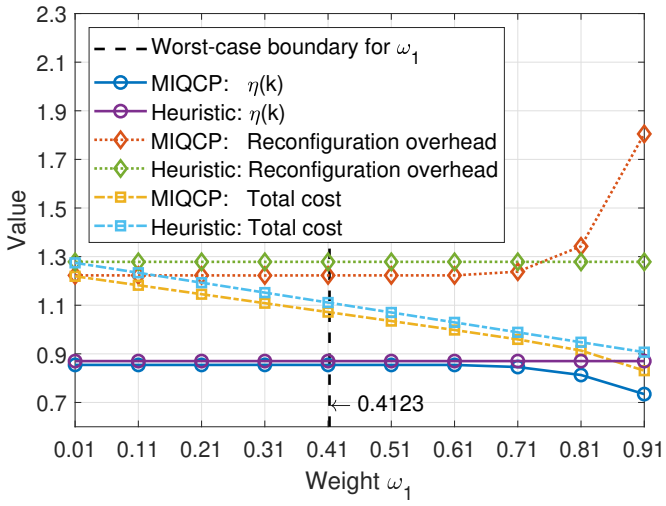


Figure 2.9: Costs with respect to weight ω_1 in objective function, for $\omega_2 = 2\omega_3$.

However, even "MMPP-1.6-0.4" for average rate in [400, 700] packet/s with flow migration performs much better than Poisson traffic arrival for average rate larger than 360 packet/s without flow migration, indicating that our flow migration model can accommodate some traffic burstiness without a significant degradation on E2E delay.

2.5.3 Comparison between MIQCP and Heuristic Solutions

Cost sensitivity to different weights

Under the 64-node network setup with three SFCs, we compare the MIQCP and heuristic solutions in terms of their cost sensitivity to different weights in (2.1). With $\omega_2 = 2\omega_3$ and $\omega_1 + \omega_2 + \omega_3 = 1$, three cost metrics including the maximum NFV node loading factor, $\eta(k)$, the reconfiguration overhead, $2N_m(k) + N_e(k)$, and the total cost, $\omega_1\eta(k) + (1 - \omega_1)(2N_m(k) + N_e(k))$, are evaluated. The first two costs are partial costs. Although the heuristic solution is in principle insensitive to ω_1, we use the same definition of total cost for a fair comparison. The three cost metrics with respect to ω_1 for both the MIQCP and heuristic solutions are given in Fig. 2.9. In both solutions, the total cost approaches to the reconfiguration overhead, for ω_1 close to 0, and approaches to the maximum NFV node loading factor, for ω_1 close to 1. For the heuristic solution, we observe constant partial

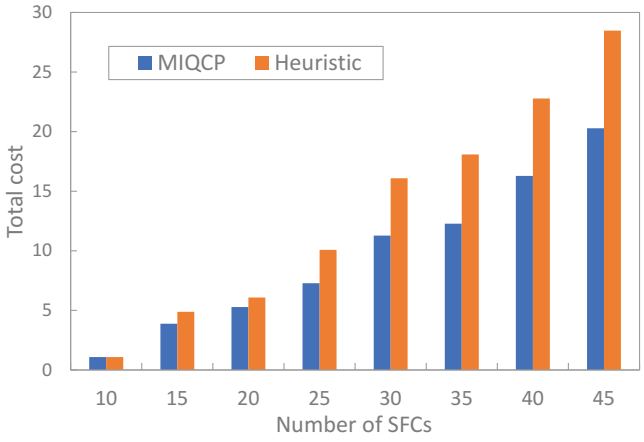


Figure 2.10: Total cost with respect to the number of SFCs.

costs with respect to ω_1, which is consistent with the design principle. For the MIQCP solution, both partial costs show a stable trend for small and medium values of ω_1 in a range larger than the theoretical worst-case range $(0, 0.4123)$. For large values of ω_1, the reconfiguration overhead of the MIQCP solution increases with ω_1, while the maximum NFV node loading factor decreases with ω_1, since much more penalization is placed on imbalanced loading than migrations.

Cost efficiency and time efficiency

We use a 256-node mesh network to compare the cost and time efficiency between the MIQCP and heuristic solutions. The comparison is performed with fixed weights in (2.1), under the condition of $\omega_1\eta_U < \omega_2$. Specifically, we have $\{\omega_1, \omega_2, \omega_3\} = \{0.4, 0.4, 0.2\}$. Eight groups of experiments are implemented with 10, 15, 20, 25, 30, 35, 40 and 45 SFCs respectively. The total cost and running time for each group are evaluated at different traffic rates from 200 to 740 packet/s. In each experiment, all SFCs have the same traffic rate, denoted by $\lambda(k)$. The initial step size, $\eta_{\Delta,0}$, and the precision, η_Δ^ϵ, are set to 0.1 and 0.0001 respectively. Fig. 2.10 shows the average total cost with respect to the number of SFCs ($|\mathcal{R}|$). It shows that the total cost obtained from both solutions increases with $|\mathcal{R}|$. Adding more SFCs tends to increase the number of overloaded NFV nodes, especially when the traffic rate is high and the added SFCs share some NFV nodes with others. Hence, more migrations tend to be triggered with more SFCs added, incurring more cost. Fig. 2.11 shows

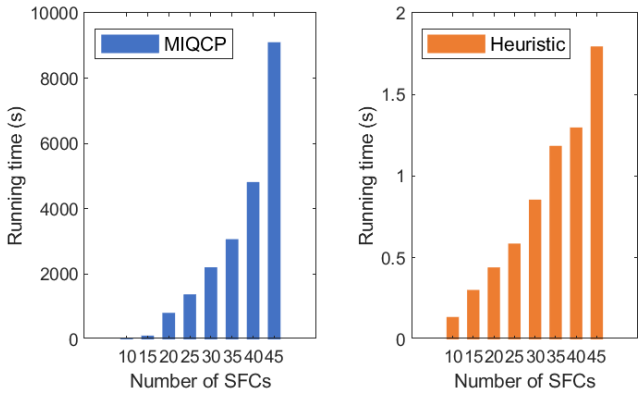


Figure 2.11: Running time with respect to the number of SFCs.

the average running time with respect to $|\mathcal{R}|$. We see an almost exponential increasing trend for the running time of the MIQCP solution. In contrast, the time complexity of the heuristic solution is much less, with a less significant increasing trend.

2.5.4 Convergence of Heuristic Algorithm

To evaluate convergence of the proposed heuristic algorithm, we plot the updating process of the threshold η_{th}, in the 45-SFC experiments with different traffic rates $\lambda(k)$, as shown in Fig. 2.12, in which $\eta(k)$ is the maximum NFV node loading factor after convergence. On each threshold updating curve corresponding to a specific traffic rate, we see that η_{th} first remains η_U due to several sequential migration decisions at the beginning and then drops with the initial step size of 0.1 until a turning point at the lower bound. After the turning point, the step size is reduced by half with each iteration until it is below the required precision 0.0001. With the increase of traffic rate to 500 packet/s, more migrations happen to gradually decouple the SFCs from each other, and more extra virtual links are observed. When the traffic rate grows larger than 500 packet/s, all SFCs are completely decoupled, with no resource sharing on NFV nodes, thus $N_m(k)$ and $N_e(k)$ are stabilized but $\eta(k)$ increases. When the traffic rate is smaller than 500 packet/s, $\eta(k)$ is close to η_U, while less migrations and extra virtual links are observed, showing a trade-off between load balancing and reconfiguration overhead.

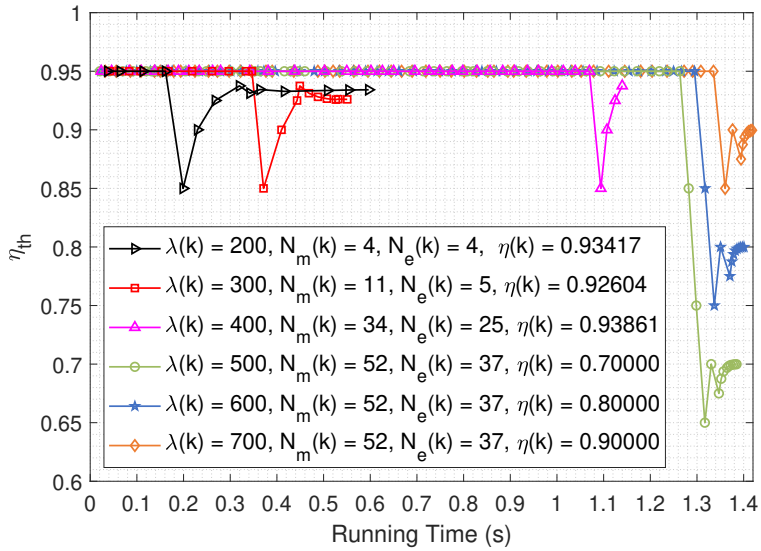


Figure 2.12: Threshold update in the heuristic algorithm, for $\eta_U = 0.95$.

2.6 Summary

In this chapter, we study a delay-aware flow migration problem for embedded services with average E2E delay requirements. A mixed integer optimization problem is formulated to address the trade-off between load balancing and reconfiguration overhead. The problem is non-convex and difficult to solve using optimization solvers. Hence, we reformulate a tractable MIQCP problem based on which the optimum of the original problem can be obtained. Numerical results show that the proposed model accommodates more traffic from services, in comparison with an SFC configuration without flow migrations. Moreover, a flow migration strategy with similar priority in load balancing and migration reduction achieves medium load balancing, as compared with flow migration strategies with a priority on either goal. Nevertheless, it achieves approximately as good performance in terms of the reconfiguration overhead as a flow migration strategy which aims at migration reduction. This result indicates the benefit of joint consideration of the two goals. A performance comparison between the MIQCP and low-complexity heuristic solutions demonstrates the effectiveness and time efficiency of the heuristic solution.

Chapter 3

Dynamic Resource Scaling for VNF over Nonstationary Traffic: A Learning Approach

In this chapter, the VNF scalability issue is studied to meet the QoS requirement in the presence of nonstationary traffic, through joint VNF migration and resource scaling. A traffic parameter learning method based on change point detection and Gaussian process regression (GPR) is proposed, to learn traffic parameters in an fBm traffic model for each stationary traffic segment within a nonstationary traffic trace. Then, the time-varying VNF resource demand is predicted from the learned traffic parameters based on an fBm resource provisioning model. With the detected change points and predicted resource demands, a VNF migration problem is formulated as an MDP with variable-length decision epochs, to maximize the long-term reward integrating load balancing, migration cost, and resource overloading penalty. A penalty-aware deep $\mathcal{Q}$-learning algorithm is proposed to incorporate awareness of resource overloading penalty, with improved performance over benchmarks in terms of training loss reduction and cumulative reward maximization.

3.1 System Model

We consider one VNF in a VNF chain, with an incoming subflow from its upstream VNF, and an outgoing subflow towards its downstream VNF. For packet processing at the VNF, it is required that the delay violation probability should not exceed an upper limit, i.e., $\mathbb{P}(\mathsf{d} > \mathsf{D}) \leq \varepsilon$, where d is a random variable denoting the experienced VNF packet processing (including queueing) delay, D is the delay bound, and ε is the maximum delay violation probability. The VNF can be placed at an NFV node in a candidate set $\mathcal{N}_C$. The considered VNF is initially placed at NFV node $n_0 \in \mathcal{N}_C$.

3.1.1 Nonstationary Traffic Model

Multi-timescale time series

Traffic arrivals at the VNF can be represented as a time series, with each traffic sample being the number of packet arrivals in non-overlapping, successive time intervals. We consider traffic arrivals in different timescales, including a medium timescale with interval length (in second) equal T_M (e.g., 20 s), and a small timescale with interval length (in second) equal T_S (e.g., 0.1 s). Let $\boldsymbol{x}_M$ denote the time series in medium timescale, given by

$$\boldsymbol{x}_M = [x_M(0), x_M(1), \cdots, x_M(i), \cdots] \tag{3.1}$$

where i (≥ 0) is an index for the medium time interval and $x_M(i)$ is the i-th traffic sample in medium timescale, representing the number of packet arrivals in the i-th medium time interval. A series of traffic samples between medium time intervals i and i' (inclusive) is given by

$$\boldsymbol{x}_M[i:i'] = [x_M(i), x_M(i+1), \cdots, x_M(i'-1), x_M(i')], \quad i' > i. \tag{3.2}$$

Similarly, a small-timescale time series is represented as

$$\boldsymbol{x}_S = [x_S(0), x_S(1), \cdots, x_S(t), \cdots] \tag{3.3}$$

where t (≥ 0) is an index for the small time interval and $x_S(t)$ is the t-th traffic sample in small timescale. Let $\boldsymbol{x}_S[t:t']$ denote a series of traffic samples between small time intervals

t and t' (inclusive), given by

$$\boldsymbol{x}_S[t:t'] = [x_S(t), x_S(t+1), \cdots, x_S(t'-1), x_S(t')], \quad t' > t. \tag{3.4}$$

Let $\mathsf{A}(t)$ denote the cumulative number of packet arrivals before small time interval t, given by

$$\mathsf{A}(t) = \begin{cases} \sum_{t'=1}^{t} x_S(t'-1), & t \geq 1 \\ 0, & t = 0. \end{cases} \tag{3.5}$$

Letting Λ be the long-term average traffic rate in packet/s, the following relationship holds:

$$\Lambda = \lim_{i' \to \infty} \frac{1}{i'} \sum_{i=0}^{i'} \frac{x_M(i)}{T_M} = \lim_{t' \to \infty} \frac{1}{t'} \sum_{t=0}^{t'} \frac{x_S(t)}{T_S} \tag{3.6}$$

where $\frac{x_M(i)}{T_M}$ and $\frac{x_S(t)}{T_S}$ are average traffic rates (in packet/s) in the i-th medium time interval and the t-th small time interval, respectively. Assume that T_M is multiples of T_S. As illustrated in Fig. 3.1, a medium-timescale time series can be mapped to a small-timescale time series within the same time duration, represented as

$$\boldsymbol{x}_M[i:i'] \Rightarrow \boldsymbol{x}_S \left[\frac{iT_M}{T_S} : \left(\frac{(i'+1)T_M}{T_S} - 1 \right) \right]. \tag{3.7}$$

Stationary traffic segments with unknown change points

The real-world network traffic usually exhibits nonstationarity [76]. Here, we consider nonstationary traffic arrivals for the VNF, as illustrated in Fig. 3.1. Assume that the nonstationary traffic time series can be partitioned into non-overlapping stationary traffic segments with unknown change points in time. Between two neighboring change points, traffic statistics such as mean and variance do not change. Let integer k (≥ 0) indicate the k-th stationary traffic segment. Consider that the change points can be located by a change point detection algorithm based on traffic statistical changes in the medium-timescale time series. Let $\mathcal{C}_M(k)$ be the index of the k-th change point in medium timescale, i.e., $x_M(\mathcal{C}_M(k))$ is the first traffic sample (in medium-timescale) of the k-th stationary traffic segment. We have $\mathcal{C}_M(0) = 0$ to indicate the beginning of the timeline. Correspondingly, the k-th change point in small timescale, $\mathcal{C}_S(k)$, is given by

$$\mathcal{C}_S(k) = \frac{\mathcal{C}_M(k) T_M}{T_S}. \tag{3.8}$$

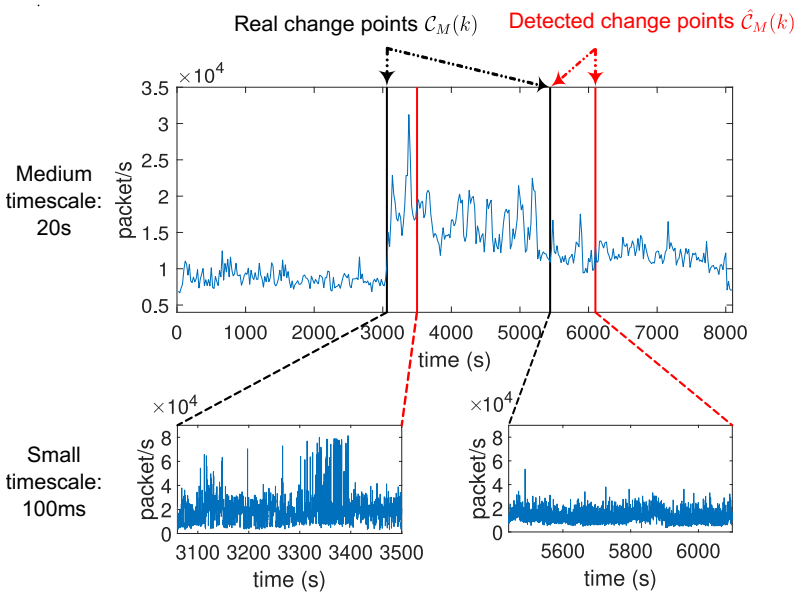


Figure 3.1: An illustration of nonstationary traffic model in different timescales.

Factional Brownian motion for a stationary traffic segment

A standard fBm process $\{Z_s(t), t = 0,1,\cdots\}$ is a centered Gaussian process with $Z_s(0) = 0$ and covariance function

$$\psi_{Z_s}(t_1, t_2) = \frac{1}{2}\left(t_1^{2H} + t_2^{2H} - |t_1 - t_2|^{2H}\right) \tag{3.9}$$

where $H \in (0,1)$ is Hurst parameter [41]. For $H \in [0.5, 1)$, the fBm process is both self-similar and LRD. A general fBm process $\{Z(t), t = 0, 1, \cdots\}$, denoting the cumulative number of packet arrivals before the t-th time unit in a stationary traffic time series, is represented by

$$Z(t) = \lambda t + \sigma Z_s(t) \tag{3.10}$$

where $\lambda = \mathbb{E}(\frac{Z(t)}{t})$ is the mean of packet arrivals in a time unit, σ is the standard deviation of packet arrivals in a time unit [41]. Here, a time unit corresponds to a small time interval. The covariance function of $Z(t)$ is given by

$$\psi_Z(t_1, t_2) = \frac{\sigma^2}{2}\left(t_1^{2H} + t_2^{2H} - |t_1 - t_2|^{2H}\right). \tag{3.11}$$

The fBm traffic model is adopted for a stationary traffic segment. For the k-th stationary traffic segment, we consider a shifted discrete timeline in small timescale, $\dot{t}$, starting at the beginning of the k-th stationary traffic segment, with $\dot{t} = t - \mathcal{C}_S(k)$. Then, we have $\dot{x}_S(\dot{t}) = x_S(t - \mathcal{C}_S(k))$, representing the number of packets arrived in the $\dot{t}$-th shifted small time interval. The cumulative number of packet arrivals in the k-th stationary traffic segment before $\dot{t}$ is modeled as an fBm process with traffic parameters $\{\lambda(k), \sigma(k), H(k)\}$, given by

$$\dot{\mathsf{A}}_k(\dot{t}) = \begin{cases} \sum_{\dot{t}'=1}^{\dot{t}} \dot{x}_S(\dot{t}' - 1), & 1 \leq \dot{t} \leq \mathcal{C}_S(k+1) - \mathcal{C}_S(k) - 1 \\ 0, & \dot{t} = 0. \end{cases} \tag{3.12}$$

3.2 Traffic Parameter Learning and Resource Demand Prediction

Since traffic statistics change across different stationary traffic segments, the amount of processing resources allocated to the VNF for probabilistic QoS guarantee, i.e., $\mathbb{P}(\mathsf{d} > \mathsf{D}) \leq \varepsilon$, should be dynamically adjusted. Here, a change-point-driven traffic parameter learning and resource demand prediction scheme is proposed, to predict resource demands from learned fBm traffic parameters of stationary traffic segments between detected change points. It provides a triggering signal for dynamic VNF migration to be discussed in Section 3.3.

3.2.1 Bayesian Online Change Point Detection

The Bayesian online change point detection (BOCPD) algorithm was first introduced in [37]. Central to the BOCPD algorithm is the run length denoted by l. A run is defined as a traffic segment with the same statistics. Online inference about the run length is performed at every time step, given a conditional prior distribution over the run length and an underlying predictive model. We use the BOCPD algorithm to detect statistical changes in mean and variance of the nonstationary medium-timescale time series $\boldsymbol{x}_M$, under the assumption that the medium-timescale traffic samples are from i.i.d Gaussian distribution $\mathcal{N}(\mu, \nu^2)$, with unknown (and perhaps changing) mean μ and standard deviation ν. A time step in the BOCPD algorithm corresponds to a medium time interval. Note that the i.i.d

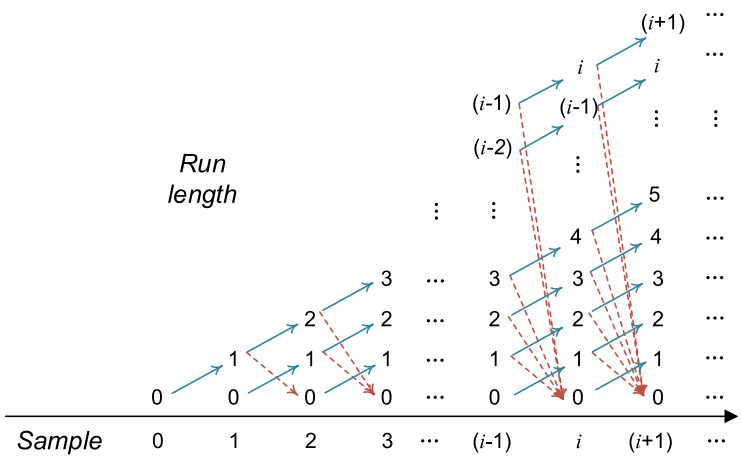


Figure 3.2: An illustration of run length growth.

Gaussian assumption is used to detect change points. For traffic parameter learning, we do not rely on such an assumption.

The run length at the i-th time step, denoted by l_i, represents the number of traffic samples before the i-th traffic sample, $x_M(i)$, within the same run. The run length l_i is a random variable taking values from $\{0, 1, \cdots, i\}$, as illustrated in Fig. 3.2. From time step $(i-1)$ to i, the run length either increases by 1 or resets to 0. For notation simplification, we omit the subscript M denoting the medium timescale, and use $\boldsymbol{x}_i$ to denote $\boldsymbol{x}_M[0:i]$. We also use $\boldsymbol{x}_i^{(l)}$ to denote $x_M[(i-l_i):i]$, which is a time series in the same run before the $(i+1)$-th traffic sample, given the run length l_i at time step i.

The joint probability of run length and observed time series at time step i, i.e., $\mathbb{P}(l_i, \boldsymbol{x}_i)$, is updated recursively from the joint probability at the previous time step, i.e., $\mathbb{P}(l_{i-1}, \boldsymbol{x}_{i-1})$, for $i \geq 1$, given by

$$\overbrace{\mathbb{P}(l_i, \boldsymbol{x}_i)}^{i\text{-th iteration}} = \sum_{l_{i-1}} \underbrace{\mathbb{P}(l_i|l_{i-1})}_{\text{conditional prior on run length}} \underbrace{\mathbb{P}(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)})}_{\text{predictive model}} \overbrace{\mathbb{P}(l_{i-1}, \boldsymbol{x}_{i-1})}^{(i-1)\text{-th iteration}}. \tag{3.13}$$

With initialization $\mathbb{P}(l_0 = 0, x_0) = 1$, for any observed value of x_0, the joint probability represents a relative likelihood. The underlying condition for (3.13) is that run length l_i is independent of $\boldsymbol{x}_i$, given l_{i-1}. The conditional prior on run length, i.e., $\mathbb{P}(l_i|l_{i-1})$, is a probability mass distribution with two outcomes, i.e., $l_i = l_{i-1} + 1$ and $l_i = 0$, as given in Appendix B. The predictive model, i.e., $\mathbb{P}(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)})$, evaluates the probability that

x_i belongs to the same run as $\boldsymbol{x}_{i-1}^{(l)}$ (i.e., $x_M[(i-1-l_{i-1}):(i-1)]$), given l_{i-1}. With a Gaussian-Inverse-Gamma prior on the unknown mean, μ, and variance, ν^2, of the i.i.d Gaussian distribution, the predictive model is described by a student-t distribution with mean $\mu_{i-1}^{(l)}$ and standard deviation $\nu_{i-1}^{(l)}$, as given in Appendix B. For each possible value of l_{i-1}, both $\mu_{i-1}^{(l)}$ and $\nu_{i-1}^{(l)}$ take different values. Through normalization, the posterior distribution of run length, $\mathbb{P}(l_i|\boldsymbol{x}_i)$, is given by

$$\mathbb{P}(l_i = i'|\boldsymbol{x}_i) = \frac{\mathbb{P}(l_i = i', \boldsymbol{x}_i)}{\sum_{l_i=0}^{i} \mathbb{P}(l_i|\boldsymbol{x}_i)}, \quad \forall i' = 0, 1, \cdots, i. \tag{3.14}$$

For traffic parameter learning and resource demand prediction, deterministic change points are required. Define the most probable run length at time step i as

$$\hat{l}_i = \text{argmax}_{l_i=\{0,\cdots,i\}} \mathbb{P}(l_i|\boldsymbol{x}_i). \tag{3.15}$$

The mean and standard deviation of the student-t predictive model corresponding to the most probable run length at time step i, i.e., $\hat{l}_i$, is seen as the estimated mean and standard deviation of the nonstationary medium-timescale time series at time step i, denoted by $\mu_i^{(\hat{l}_i)}$ and $\nu_i^{(\hat{l}_i)}$ respectively. Time step i is identified as a change point if the following two conditions are satisfied. First, the gap between the most probable run lengths at time steps $(i-1)$ and i, i.e., $\hat{l}_{i-1}$ and $\hat{l}_i$, is larger than a threshold Υ_l, given by

$$\hat{l}_{i-1} - \hat{l}_i > \Upsilon_l; \tag{3.16}$$

Second, the normalized absolute difference between the estimated mean plus standard deviation at time step i and $(i-1)$ is beyond a predefined threshold Υ_d, given by

$$\frac{\left|\left(\mu_i^{(\hat{l}_i)} + \nu_i^{(\hat{l}_i)}\right) - \left(\mu_{i-1}^{(\hat{l}_{i-1})} + \nu_{i-1}^{(\hat{l}_{i-1})}\right)\right|}{\mu_{i-1}^{(\hat{l}_{i-1})} + \nu_{i-1}^{(\hat{l}_{i-1})}} > \Upsilon_d. \tag{3.17}$$

The k-th detected change point, denoted by $\hat{\mathcal{C}}_M(k)$, is an estimated value of the real change point $\mathcal{C}_M(k)$, i.e., the index of the first medium-timescale traffic sample in the k-th stationary traffic segment. The BOCPD algorithm has a linear space and time complexity per time-step in the number of medium-timescale traffic samples after the previously detected change point [37]. The stochastic BOCPD method results in a latency between $\mathcal{C}_M(k)$ and $\hat{\mathcal{C}}_M(k)$, as illustrated in Fig. 3.1, in which the real and detected change points are indicated by the black and red vertical lines, respectively. The latency cannot be avoided since it is inherent to the BOCPD algorithm. We exploit the latency for a *look-back* traffic parameter learning.

3.2.2 Traffic Parameter Learning

Let i_0 be a small integer[1] such that (i_0-1) medium-timescale traffic samples before the $\hat{\mathcal{C}}_M(k)$-th one belong to the k-th stationary traffic segment. The i_0 medium-timescale traffic samples including the $\hat{\mathcal{C}}_M(k)$-th one correspond to $\frac{i_0 T_M}{T_S}$ small-timescale traffic samples within the same time duration, given by

$$\begin{aligned}\boldsymbol{x}_M[(\hat{\mathcal{C}}_M(k)-i_0+1):\hat{\mathcal{C}}_M(k)] &\Rightarrow \boldsymbol{x}_S\left[\frac{(\hat{\mathcal{C}}_M(k)-i_0+1)T_M}{T_S}:\left(\frac{(\hat{\mathcal{C}}_M(k)+1)T_M}{T_S}-1\right)\right] \\ &= \boldsymbol{x}_S\left[\hat{\mathcal{C}}_S(k):\left(\hat{\mathcal{C}}_S(k)+\frac{i_0 T_M}{T_S}-1\right)\right]\end{aligned} \tag{3.18}$$

where $\hat{\mathcal{C}}_S(k)=\frac{(\hat{\mathcal{C}}_M(k)-i_0+1)T_M}{T_S}$ is the estimated k-th change point in small timescale. The $\frac{i_0 T_M}{T_S}$ traffic samples are used to learn fBm traffic parameters of the k-th stationary traffic segment. Compared with a *look-ahead* counterpart, the *look-back* mechanism avoids another latency after the detected change point, for collecting sufficient traffic samples.

We consider a modified shifted discrete timeline, $\tilde{t}$, with $\tilde{t}=t-\hat{\mathcal{C}}_S(k)$, for the k-th stationary traffic segment. Correspondingly, we have $\tilde{x}_S(\tilde{t})=x_S(t-\hat{\mathcal{C}}_S(k))$, representing the number of packets arrived in the $\tilde{t}$-th modified shifted small time interval. The cumulative number of packet arrivals in the k-th stationary traffic segment before $\tilde{t}$, is given by

$$\tilde{\mathsf{A}}_k(\tilde{t})=\begin{cases}\sum_{\tilde{t}'=1}^{\tilde{t}}\tilde{x}_S(\tilde{t}'-1), & 1\le\tilde{t}\le\hat{C}_S(k+1)-\hat{C}_S(k)-1\\ 0, & \tilde{t}=0.\end{cases} \tag{3.19}$$

We use $\{\tilde{\mathsf{A}}_k(\tilde{t}), 0\le\tilde{t}\le\frac{i_0 T_M}{T_S}-1\}$ to learn fBm traffic parameters of the k-th stationary traffic segment. Consider the following Gaussian process regression (GPR) model

$$\tilde{\mathsf{A}}_k(\tilde{t})\sim\mathcal{GP}(\lambda(k)\tilde{t},\psi_k(\tilde{t}_1,\tilde{t}_2)) \tag{3.20}$$

where $\lambda(k)\tilde{t}$ is the mean function and $\psi_k(\tilde{t}_1,\tilde{t}_2)$ is the fBm covariance function given by

$$\psi_k(\tilde{t}_1,\tilde{t}_2)=\frac{\sigma^2(k)}{2}\left(\tilde{t}_1^{2H(k)}+\tilde{t}_2^{2H(k)}-|\tilde{t}_1-\tilde{t}_2|^{2H(k)}\right). \tag{3.21}$$

The fBm traffic parameters $\{\lambda(k),\sigma(k),H(k)\}$ are referred to as hyper-parameters in the GPR framework [40, 77]. Let $\boldsymbol{t}_k=[0,1,\cdots,(\frac{i_0 T_M}{T_S}-1)]$ be training inputs and $\mathbf{A}_k=$

[1]The value of i_0 should be smaller than the minimum value of the most probable run lengths at any detected change points.

$[\tilde{\mathsf{A}}_k(0), \tilde{\mathsf{A}}_k(1), \cdots, \tilde{\mathsf{A}}_k(\frac{i_0 T_M}{T_S} - 1)]$ be training outputs. Then, we have the following joint Gaussian distribution

$$\mathbf{A}_k \sim \mathcal{N}(\lambda(k)\boldsymbol{t}_k, \Psi_k) \tag{3.22}$$

where Ψ_k is a $\frac{i_0 T_M}{T_S}$-by-$\frac{i_0 T_M}{T_S}$ covariance matrix, with $\Psi_k(i,j) = \psi_k(i,j)$. The GPR model is trained, i.e., the hyper-parameters are learned, by maximizing the following log-marginal likelihood function with a gradient optimizer

$$\begin{aligned} &\log \mathbb{P}(\mathbf{A}_k \,|\, \boldsymbol{t}_k; \{\lambda(k), \sigma(k), H(k)\}) = \\ &-\frac{1}{2}\left[(\mathbf{A}_k - \lambda(k)\boldsymbol{t}_k)^T \Psi_k^{-1} (\mathbf{A}_k - \lambda(k)\boldsymbol{t}_k) + \log|\Psi_k| + \frac{i_0 T_M}{T_S}\log 2\pi\right]. \end{aligned} \tag{3.23}$$

For the GPR-based traffic parameter learning with $\frac{i_0 T_M}{T_S}$ traffic samples, it has $\mathcal{O}(\frac{i_0 T_M}{T_S}^3)$ time complexity and $\mathcal{O}(\frac{i_0 T_M}{T_S}^2)$ space complexity due to the inversion of a covariance matrix in (3.23) [78]. Such a complexity is feasible on a desktop computer for dataset sizes up to a few thousands. There are sparse approximation algorithms to reduce the complexity of Gaussian process regression [78].

To evaluate the learning accuracy, one-step-ahead predictions for t_0 subsequent small time intervals are performed using the trained GPR model. The one-step-ahead prediction at time $\tilde{t}$ ($\frac{i_0 T_M}{T_S} - 1 \le \tilde{t} \le \frac{i_0 T_M}{T_S} + t_0 - 2$) is to predict $\tilde{\mathsf{A}}_k(\tilde{t}^*)$, given $\tilde{t}^* = \tilde{t} + 1$ and a set of observed data $\mathcal{D} = (\boldsymbol{t}, \mathbf{A})$ with $\boldsymbol{t} = [0, 1, \cdots, \tilde{t}]$ and $\mathbf{A} = [\tilde{\mathsf{A}}_k(0), \tilde{\mathsf{A}}_k(1), \cdots, \tilde{\mathsf{A}}_k(\tilde{t})]$). The GPR gives a Gaussian posterior distribution of $\tilde{\mathsf{A}}_k(\tilde{t}^*)$ conditioned on $\tilde{t}^*$ and $\mathcal{D}$, as

$$\mathbb{P}(\tilde{\mathsf{A}}_k(\tilde{t}^*) | \tilde{t}^*, \mathcal{D}) \sim \mathcal{N}(\mu_{\mathcal{GP};k}(\tilde{t}^*), \sigma^2_{\mathcal{GP};k}(\tilde{t}^*)) \tag{3.24}$$

with

$$\begin{cases} \mu_{\mathcal{GP};k}(\tilde{t}^*) = \boldsymbol{\psi}_k(\boldsymbol{t}, \tilde{t}^*)^T (\Psi)^{-1} \mathbf{A} \\ \sigma^2_{\mathcal{GP};k}(\tilde{t}^*) = \psi_k(\tilde{t}^*, \tilde{t}^*) - \boldsymbol{\psi}_k(\boldsymbol{t}, \tilde{t}^*)^T (\Psi)^{-1} \boldsymbol{\psi}_k(\boldsymbol{t}, \tilde{t}^*). \end{cases} \tag{3.25}$$

In (3.25), $\boldsymbol{\psi}_k(\boldsymbol{t}, \tilde{t}^*)$ is a $(\tilde{t}+1)$-by-1 vector with the $\mathtt{j}$-th component equal to $\psi_k(\mathtt{j}, \tilde{t}^*)$, and Ψ is a $(\tilde{t}+1)$-by-$(\tilde{t}+1)$ covariance matrix with $\Psi(\mathtt{j}, \mathtt{j}') = \psi_k(\mathtt{j}, \mathtt{j}')$. The mean, $\mu_{\mathcal{GP};k}(\tilde{t}^*)$, is taken as a point estimate for the prediction output, and the variance, $\sigma^2_{\mathcal{GP};k}(\tilde{t}^*)$, provides an uncertainty measure for the point estimate. With the predictive distribution for $\tilde{\mathsf{A}}_k(\tilde{t}^*)$, the traffic sample in time interval $\tilde{t}^*$, i.e., $\tilde{x}_S(\tilde{t}^*)$, is predicted as

$$\hat{\tilde{x}}_S(\tilde{t}^*) = \mu_{\mathcal{GP};k}(\tilde{t}^*) - \tilde{\mathsf{A}}_k(\tilde{t}^* - 1). \tag{3.26}$$

The prediction error of the t_0 traffic samples in the k-th stationary traffic segment, $\mathtt{E}_k$, is defined as the normalized root-mean-squared deviation between the t_0 predicted traffic samples and the corresponding ground truth, given by

$$\mathtt{E}_k = \frac{\sqrt{\sum_{\tilde{t}^*=\frac{i_0 T_M}{T_S}}^{\frac{i_0 T_M}{T_S}+t_0-1} \left(\hat{\tilde{x}}_S(\tilde{t}^*) - \tilde{x}_S(\tilde{t}^*)\right)^2}}{(x_S^{max} - x_S^{min})\sqrt{t_0}}. \tag{3.27}$$

The normalization constant is the scale of small-timescale traffic samples, i.e., $(x_S^{max} - x_S^{min})$. A smaller $\mathtt{E}_k$ value indicates a higher learning accuracy for traffic parameters.

3.2.3 Resource Demand Prediction

With the learned fBm traffic parameters, the resource demand of the k-th stationary traffic segment can be predicted. Consider an fBm traffic input with parameters $\{\lambda, \sigma, H\}$ to an infinite buffer, with a constant service rate of R packets per small time interval. The buffer overflow probability, i.e., the probability that queue length q is beyond a threshold q_B, is approximately given by [41, 79]

$$\mathbb{P}(q > q_B) \simeq \exp\left(-\inf_{t\geq 0} \frac{[q_B + (\mathsf{R}-\lambda)t]^2}{2\sigma^2 t^{2H}}\right) \tag{3.28}$$

which has been shown accurate even for a small value of q_B by simulation studies. Correspondingly, the delay violation probability can be approximated by

$$\mathbb{P}(\mathsf{d}_S > \mathsf{D}_S) \simeq \exp\left(-\inf_{t\geq 0} \frac{[\mathsf{R}\mathsf{D}_S + (\mathsf{R}-\lambda)t]^2}{2\sigma^2 t^{2H}}\right) \tag{3.29}$$

where $\mathsf{d}_S = \frac{\mathsf{d}}{T_S}$ is the random VNF packet processing delay in number of small time intervals, and $\mathsf{D}_S = \frac{\mathsf{D}}{T_S}$ is the corresponding delay bound. To provide probabilistic QoS guarantee (i.e., $\mathbb{P}(\mathsf{d}_S > \mathsf{D}_S) \leq \varepsilon$) to the VNF with minimum resources, we should find

$$\min\left\{\mathsf{R} \mid \forall t \geq 0, [\mathsf{R}\mathsf{D}_S + (\mathsf{R}-\lambda)t]^2 \geq (-2\log\varepsilon)\sigma^2 t^{2H}\right\} \tag{3.30}$$

which leads to

$$\mathsf{R}_{min} = \sup_{t\geq 0} \frac{\lambda t + \sqrt{-2\log\varepsilon}\,\sigma t^H}{t + \mathsf{D}_S}. \tag{3.31}$$

The value of t achieving the supremum can be obtained by setting the derivative of R_{min} with respect to t to zero, i.e.,

$$\frac{\sqrt{-2\log\varepsilon}\sigma\mathsf{D}_S H t^{H-1} + \sqrt{-2\log\varepsilon}\sigma(H-1)t^H + \lambda\mathsf{D}_S}{(t+\mathsf{D}_S)^2} = 0. \tag{3.32}$$

With the fBm resource provisioning model given in (3.31), the predicted resource demand (in packet/s) of the k-th stationary traffic segment, denoted by $\mathsf{R}(k)$, is calculated from the learned fBm traffic parameters, i.e., $\{\lambda(k), \sigma(k), H(k)\}$, the QoS requirement, and the small time interval length, i.e., T_S.

3.3 Deep Reinforcement Learning for Dynamic VNF Migration

The BOCPD algorithm locates the prior-unknown change points of the nonstationary traffic, which determines the boundaries between consecutive stationary traffic segments. They are also boundaries between consecutive decision epochs (with variable lengths) for VNF scaling and necessary VNF migrations. The length of decision epoch k is equal to $(\hat{\mathcal{C}}_M(k+1) - \hat{\mathcal{C}}_M(k))T_M$. Once change point $\hat{\mathcal{C}}_M(k)$ is detected, the resource demand $\mathsf{R}(k)$ of the upcoming k-th stationary traffic segment is predicted, based on which a VNF migration decision is made.

3.3.1 VNF Migration Problem Formulation

For VNF migration, we jointly consider the migration cost and load balancing. Let $\{\mathrm{a}_k^n, n \in \mathcal{N}_C\}$ be a binary variable set, with $\mathrm{a}_k^n = 1$ if the VNF is placed at NFV node n during the k-th decision epoch, and $\mathrm{a}_k^n = 0$ otherwise. Let a_k $(0 \le \mathrm{a}_k \le |\mathcal{N}_C| - 1)$ be an integer denoting the VNF location during decision epoch k, with $\mathrm{a}_k = n$ if the VNF is placed at NFV node n. The relationship between $\{\mathrm{a}_k^n\}$ and a_k is given by

$$\mathrm{a}_k^n = \begin{cases} 1, & \text{if } \mathrm{a}_k = n \\ 0, & \text{otherwise.} \end{cases} \tag{3.33}$$

Define the background resource loading factor of NFV node n during decision epoch k, denoted by $\eta_n^B(k)$, as the average ratio between the amount of processing resources (in

packet/s) allocated to background traffic at NFV node n during decision epoch k and the processing resource capacity R_n (in packet/s) of NFV node n. The resource loading factor of NFV node n during decision epoch k, denoted by $\eta_n(k)$, is dependent on both $\eta_n^B(k)$ and VNF placement, given by

$$\eta_n(k) = \eta_n^B(k) + \frac{\mathrm{a}_k^n \mathsf{R}(k)}{\mathsf{R}_n}. \tag{3.34}$$

The cost for imbalanced loading during decision epoch k is defined as the maximum resource loading factor among all NFV nodes in $\mathcal{N}_C$, given by

$$c_k^{(1)} = \max_{n \in \mathcal{N}_C} \eta_n(k), \tag{3.35}$$

since minimizing $c_k^{(1)}$ achieves load balancing among all the candidate NFV nodes. Assume that each VNF migration incurs the same migration cost. Then, we can use the total number of migrations to denote the total migration cost, given by

$$c_k^{(2)} = \begin{cases} \sum_{n \in \mathcal{N}_C} \sum_{n' \in \mathcal{N}_C \setminus n} \mathrm{a}_{k-1}^n \mathrm{a}_k^{n'}, & \text{if } k > 0 \\ 0, & \text{if } k = 0 \end{cases} \tag{3.36}$$

where a_{k-1}^n is a known value at decision epoch k (> 0). In the single VNF scenario, we have $c_k^{(2)} = 1$ for $k > 0$ if the VNF placement changes from decision epoch $(k-1)$ to decision epoch k, and $c_k^{(2)} = 0$ otherwise. The total cost is a weighted combination of the two costs, given by

$$c_k = \omega^{(\mathsf{C})} c_k^{(1)} + (1 - \omega^{(\mathsf{C})}) c_k^{(2)} \tag{3.37}$$

where $\omega^{(\mathsf{C})}$ is a weighting factor in $(0, 1)$. In stepwise optimization for cost minimization in the short term, total cost c_k is minimized at each decision epoch k, subject to processing resource capacity constraints at the NFV nodes, i.e., the resource loading factors of all NFV nodes should not exceed 1. For cost minimization in the long run, the VNF migration problem can be formulated as an MDP, with the state, action, and reward defined as follows:

- *State* – At decision epoch k, the state is composed of four parts: the k-th change point, the predicted resource demand of the k-th stationary traffic segment, the background resource loading factors of all candidate NFV nodes during decision epoch k,

and the previous VNF placement a_{k-1}. Thus, the state for decision epoch k is represented as $\boldsymbol{s}_k = [\hat{\mathcal{C}}(k), \mathsf{R}(k), \{\eta_n^B(k)\}, a_{k-1}]$. Here, $\hat{\mathcal{C}}(k)$ is a real number representing the k-th estimated change point in hour, given by

$$\hat{\mathcal{C}}(k) = \frac{\hat{\mathcal{C}}_M(k) T_M}{3600} \mod 24 \tag{3.38}$$

where the modulo operation limits $\hat{\mathcal{C}}(k)$ in $[0, 24)$;

- *Action* – The action at decision epoch k is the new VNF placement, i.e., a_k. We use a_k instead of $\{\mathrm{a}_k^n\}$ as the action to limit the dimensionality of action space;

- *Reward* – In an unconstrained MDP, the violation of resource capacity constraints is penalized by an extra term in reward. Hence, the reward for decision epoch k is

$$\mathrm{r}_k = -\left(c_k + c^{(\mathsf{P})} \mathsf{f}_k^{(\mathsf{P})}\right) \tag{3.39}$$

where c_k is the total cost for VNF migration at decision epoch k as given in (3.37), $\mathsf{f}_k^{(\mathsf{P})}$ is a binary flag indicating whether there is penalty due to resource overloading, and $c^{(\mathsf{P})}$ is a constant representing the level of penalty. Assume that resource overloading is only due to improper VNF placement, i.e., the background traffic does not overload the NFV nodes ($\eta_n^B(k) < 1$). Then, the penalty flag is defined as where

$$\mathsf{f}_k^{(\mathsf{P})} = \begin{cases} 1, & c_k^{(1)} > \eta_U \\ 0, & \text{otherwise} \end{cases} \tag{3.40}$$

where η_U ($0 < \eta_U \leq 1$) is an upper limit for the maximum resource loading factor without penalty. In practice, we select η_U as a number close to but smaller than 1, e.g., $\eta_U = 0.95$, to penalize loading factors close to 1, with the consideration that the penalty cannot be completely avoided in a learning-based solution due to exploration. Moreover, if the predicted resource demand is very large, it is possible that there is no feasible VNF placement without resource overloading. A potential solution is to throttle the traffic when resource overloading is foreseen to happen. Here, we assume that the VNF placement without resource overloading is always feasible and do not consider traffic throttling.

Algorithm 4: Penalty-aware deep $\mathcal{Q}$-learning

Initialize: Evaluation and target D$\mathcal{Q}$Ns with random weights, set learning parameters as listed in Table 3.2.
for *each episode* **do**
  Initialize VNF placement at NFV node n_0.
  for *each learning step (decision epoch)* **do**
    Observe current state $\boldsymbol{s}_k$, select an action a_k according to the ϵ-greedy policy in (3.42).
    Execute action a_k, collect reward r_k and penalty flag $\mathsf{f}_k^{(\mathsf{P})}$, and see the next state $\boldsymbol{s}_{k+1}$.
    Store transition $(\boldsymbol{s}_k, \mathrm{a}_k, \mathrm{r}_k, \mathsf{f}_k^{(\mathsf{P})}, \boldsymbol{s}_{k+1})$ into replay memory, with initial priority $p_k = \max_{j<k} p_j$.
    for J *iterations* **do**
      Sample a transition $(\boldsymbol{s}_j, \mathrm{a}_j, \mathrm{r}_j, \mathsf{f}_j^{(\mathsf{P})}, \boldsymbol{s}_{j+1})$ with probability $\mathbb{P}(j)$.
      Compute importance-sampling weight $\mathtt{w}_j$.
      Compute target value $\mathtt{y}_j$ and TD error δ_j.
      Update transition priority p_j.
      Perform a gradient descent, i.e., $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\mathcal{Q}} (\mathtt{w}_j \delta_j) \nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{s}_j, \mathrm{a}_j)$.
    end
    Decrease the exploration probability ϵ_2 by a step $\epsilon_{2,\Delta}$, if $\epsilon_2 > \epsilon_2^{min}$.
    Every $K_{\boldsymbol{\theta}}$ steps, set $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$.
  end
end
Output: Trained evaluation and target D$\mathcal{Q}$Ns.

3.3.2 Penalty-Aware Deep Q-Learning Algorithm

We solve the MDP by an RL approach, when transition probabilities among states are unavailable. Consider an episodic task, in which an RL agent interacts with the VNF migration environment in a sequence of episodes, with a finite number of learning steps in each episode. Here, a learning step corresponds to a decision epoch, and an episode corresponds to a time duration such as one day, one week, or one month. At the beginning of an episode, the VNF placement is initialized at NFV node n_0. Within an episode, an agent observes state $\boldsymbol{s}_k$ and takes action a_k at the beginning of decision epoch k. At the end of decision epoch k, the agent receives reward r_k, and sees new state $\boldsymbol{s}_{k+1}$. The goal is to find a policy, $\pi(\boldsymbol{s})$, mapping a state to an action, to maximize the expected cumulative (episodic) discounted reward $\mathbb{E}(\sum_{k=0}^{K-1}\gamma^k \mathrm{r}_k)$, where K is the number of variable-length decision epochs in an episode, and $\gamma \in (0,1]$ is the discount factor. In $\mathcal{Q}$-learning [42], a state-action value function $\mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}_k)$ is defined as

$$\mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}_k) = \mathbb{E}\left[\sum_{k'=k}^{K-1} \gamma^{k'-k} r_{k'} | \boldsymbol{s}_k, \mathrm{a}_k\right] \tag{3.41}$$

The $\mathcal{Q}$-learning is an off-policy algorithm adopting the ϵ-greedy policy

$$\pi(\boldsymbol{s}_k) = \begin{cases} \underset{\mathrm{a}}{\text{argmax}}\ \mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}), & \text{with probability } (1-\epsilon_2) \\ \text{random action}, & \text{with probability } \epsilon_2 \end{cases} \tag{3.42}$$

where ϵ_2 is the exploration probability. We use a gradually decreasing ϵ_2 from 1 to a minimum value ϵ_2^{min}, with a step size $\epsilon_{2,\Delta}$, to transit smoothly from exploration to exploitation.

The formulated MDP is featured by a high-dimensional combinational state space and a low-dimensional discrete action space. To tackle the curse of dimensionality, deep $\mathcal{Q}$-learning adopts two deep Q-networks (DQNs) with the same neural network structure as $\mathcal{Q}$ function approximators, i.e., evaluation D$\mathcal{Q}$N ($\mathcal{Q}$) with weights $\boldsymbol{\theta}$ and target D$\mathcal{Q}$N ($\hat{\mathcal{Q}}$) with slowly updated weights $\hat{\boldsymbol{\theta}}$ [80]. Every $K_{\boldsymbol{\theta}}$ learning steps, $\hat{\boldsymbol{\theta}}$ is replaced by $\boldsymbol{\theta}$. The policy in (3.42) is based on evaluation D$\mathcal{Q}$N, which is trained by minimizing a loss function $\mathbb{L}(\boldsymbol{\theta})$, given by

$$\mathbb{L}(\boldsymbol{\theta}) = \mathbb{E}\left[(\mathrm{y}_k - \mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}_k; \boldsymbol{\theta}))^2\right] \tag{3.43}$$

through gradient descent on $\boldsymbol{\theta}$, where y_k is a target value estimated by target D$\mathcal{Q}$N:

$$\mathrm{y}_k = \mathrm{r}_k + \gamma \max_{\mathrm{a}} \hat{\mathcal{Q}}(\boldsymbol{s}_{k+1}, \mathrm{a}; \hat{\boldsymbol{\theta}}). \tag{3.44}$$

If an episode terminates at the k-th learning step, $\mathtt{y}_k$ is set as r_k. A gradient descent on $\boldsymbol{\theta}$ is performed by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{1}{2}\alpha_{\mathcal{Q}} \nabla_{\boldsymbol{\theta}} \mathbb{L}(\boldsymbol{\theta}) = \boldsymbol{\theta} + \alpha_{\mathcal{Q}} \delta_k \nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}_k) \tag{3.45}$$

where $\alpha_{\mathcal{Q}}$ is the learning rate, and $\delta_k = \mathtt{y}_k - \mathcal{Q}(\boldsymbol{s}_k, \mathrm{a}_k; \boldsymbol{\theta})$ is the temporal-difference (TD) error.

Experience replay is introduced in deep $\mathcal{Q}$-learning for stable convergence [80]. At each learning step, $\boldsymbol{\theta}$ is updated with a mini-batch (size equal J of experiences $(\boldsymbol{s}_j, \mathrm{a}_j, \mathrm{r}_j, \boldsymbol{s}_{j+1})$ uniformly sampled from a replay memory. Experience replay breaks the temporal correlation among experiences, and liberates RL agents from learning with transitions in the same order as they appear. Prioritized experience replay achieves more learning efficiency through further liberating RL agents from considering transitions in the same frequency as they appear [81, 82]. It assigns a priority, p_j, for transition j sampled from the replay memory, which is the magnitude of TD error δ_j plus a very small value ϵ_1. The sampling probability of transition j is

$$\mathbb{P}(j) = \frac{p_j^{o_1}}{\sum_{j=1}^{\mathtt{M}} p_j^{o_1}} \tag{3.46}$$

where $\mathtt{M}$ is the size of replay memory and o_1 determines the level of prioritization.

In the VNF migration problem, it is desired that the deep $\mathcal{Q}$-learning algorithm converges to a solution without resource overloading penalty in the whole episode. However, such experiences are rare at the early learning stage with a lot of exploration, especially if an episode contains a large number of transitions. To learn more from such rare desired experiences, we extend the prioritized experience replay technique to consider penalty-awareness. Among the original prioritized transitions with high absolute TD errors, we place more priority on those transitions with zero penalty, given by

$$p_j = \omega^{(\mathtt{TD})} |\delta_j| + (1 - \omega^{(\mathtt{TD})})(1 - \mathsf{f}_j^{(\mathsf{P})}) + \epsilon_1 \tag{3.47}$$

where $\omega^{(\mathtt{TD})} \in [0, 1]$ is a parameter controlling the relative importance of TD error and penalty avoidance, and $0 < \epsilon_1 \ll 1$ is a very small constant. In practice, we select $\omega^{(\mathtt{TD})}$ close to 1, e.g., 0.99, to incorporate penalty-awareness without significant degradation on convergence speed. Correspondingly, at every learning step, the penalty flag, $\mathsf{f}_j^{(\mathsf{P})}$ in (3.40), is recorded, and a five-tuple transition $(\boldsymbol{s}_j, \mathrm{a}_j, \mathrm{r}_j, \mathsf{f}_j^{(\mathsf{P})}, \boldsymbol{s}_{j+1})$ instead of the original four-tuple

Table 3.1: Traffic sets with different randomness levels

Traffic set	Change points	Resource demands
1	Detected	Predicted
2	Detected $\pm$ $[0, 0.1]$ hour	Predicted
3	Detected	Predicted $\pm$ $[0\%, 5\%]$
4	Detected $\pm$ $[0, 0.1]$ hour	Predicted $\pm$ $[0\%, 5\%]$

Table 3.2: List of parameters in deep $\mathcal{Q}$-learning

$\alpha_{\mathcal{Q}}$	Learning rate	10^{-6}
γ	Discount factor	0.9
ϵ_2^{min}	Minimum exploration probability	0.01
$\epsilon_{2,\Delta}$	Step size of exploration probability	5×10^{-6}
$K_{\boldsymbol{\theta}}$	Number of steps to replace $\hat{\boldsymbol{\theta}}$ by $\boldsymbol{\theta}$	200
$\mathtt{M}$	Memory size	2000
J	Batch size	200
$\omega^{(\mathtt{TD})}$	Weight in the priority	0.99

transition, $(\boldsymbol{s}_j, \mathrm{a}_j, \mathrm{r}_j, \boldsymbol{s}_{j+1})$, is stored in the replay memory, for priority calculation based on (3.47). A deep $\mathcal{Q}$-learning algorithm with penalty-aware prioritized experience replay is presented in Algorithm 4. The prioritization leads to a loss of diversity, which can be corrected with an importance-sampling weight $\mathtt{w}_j$, given by [82]

$$\mathtt{w}_j = (B \cdot \mathbb{P}(j))^{-o_2} \,/\, \max_{j'}(\mathtt{w}_{j'}) \tag{3.48}$$

where o_2 controls the level of compensation. The TD error δ_j is replaced by a weighted TD error $\mathtt{w}_j\delta_j$ in a gradient descent step with transition j, as given in line 13 of Algorithm 4.

3.4 Performance Evaluation

We use a real-world backbone traffic trace from the MAWI working group of the WIDE project for performance evaluation, which provides packet-level information collected from Internet backbone links [20]. We select two most recent 48-hour-long traces collected

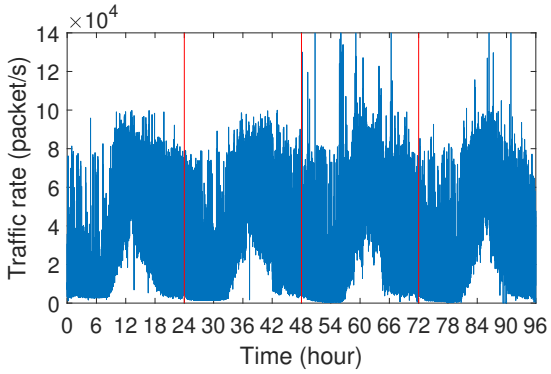


Figure 3.3: The extracted HTTP trace trace in four days.

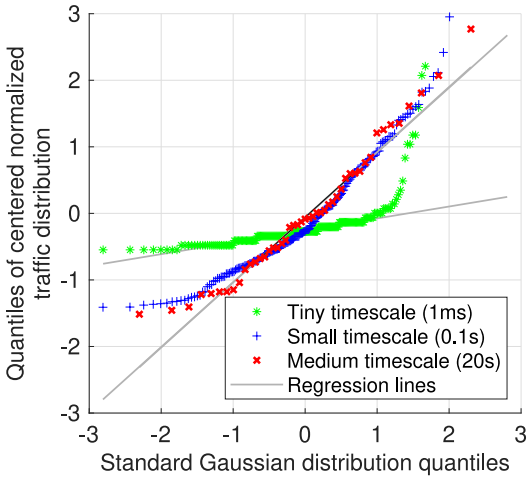


Figure 3.4: Quantile-quantile (QQ) plots for different timescales.

from the transit link of WIDE backbone connecting the upstream ISP. The specific days are 2018/05/09, 2018/05/10, 2019/04/09, and 2019/04/10. We extract the Hypertext Transfer Protocol (HTTP) traffic from port 443 as an aggregate service flow, and select $T_S = 0.1$ s and $T_M = 20$ s as the lengths of small and medium time intervals, respectively. Fig. 3.3 shows the four-day HTTP traffic trace in small timescale, which exhibits a daily periodic traffic pattern. The quantile-quantile (QQ) plots for the distribution of centered normalized number of packet arrivals (with mean equal to 0 and standard deviation equal to 1) in different timescales (within the same stationary traffic segment) versus a standard Gaussian distribution are presented in Fig. 3.4. It shows that the traffic distributions in both small and medium timescales are approximately Gaussian with heavy tails. The traffic distribution in a tiny timescale (1 ms) is more bursty and completely not Gaussian due to insufficient aggregation of packet arrivals in each tiny time interval. The two thresholds in change point detection, i.e., Υ_l and Υ_d, are set as 10 and 5%, respectively. We select $i_0 = 4$ to have 800 small-timescale traffic samples in each stationary traffic segment for traffic parameter learning, which gives high computation efficiency while achieving a good accuracy. For each daily traffic trace, change points are detected, and resource demands are predicted for the identified stationary traffic segments. There are 25, 20, 26, and 24 detected change points in the four daily traffic traces, respectively. The accuracy of traffic parameter learning is evaluated with $t_0 = 1000$ small-timescale traffic samples.

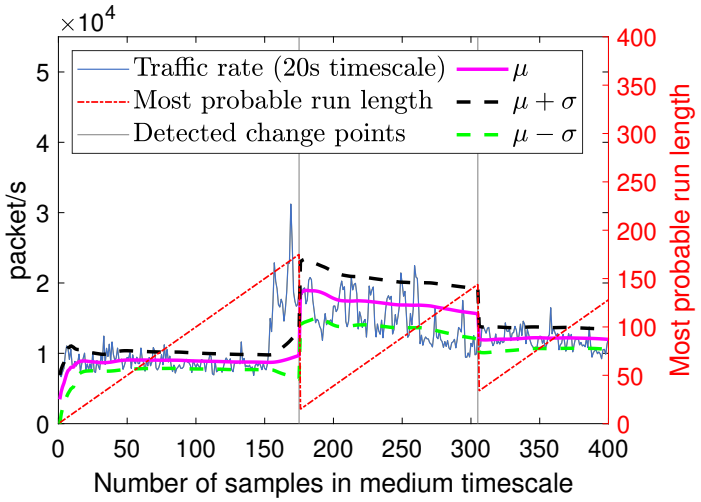


Figure 3.5: Results of change point detection for a nonstationary traffic segment.

For dynamic VNF migration, we consider one VNF initially placed at NFV node n_0, with another five candidate NFV nodes located in its neighborhood. All NFV nodes have the same processing capacity $\mathsf{R}_n = 125000$ packet/s. The background resource loading factor of each NFV node varies between 10% and 90%, in different patterns with peaks at different time in a day. We set weighting factor $\omega^{(\mathsf{C})} = 0.6$ and penalty level $c^{(\mathsf{P})} = 5$. For deep $\mathcal{Q}$-learning, an episode corresponds to one week, to have sufficient learning steps (decision epochs) with periodic dynamics in both change points and resource demands within an episode. The weekly traffic in one episode is artificially composited by daily traffic of the four days in random order, with different randomness levels in both change points and resource demands, as described in Table 3.1. The randomness level (in hour) around change points follows a uniform distribution in $[0, 0.1]$, and the randomness level around resource demands follows a uniform distribution in $[0\%, 5\%]$. We use a D$\mathcal{Q}$N structure with one hidden layer of 20 neurons and Relu as the activation function, with important learning parameters summarized in Table 3.2.

Fig. 3.5 shows results of change point detection for a nonstationary traffic segment in 8000 s, corresponding to 400 medium time intervals. A zigzag trend is observed for the most probable run length. The detected change points are indicated by gray vertical lines. Online estimation of mean and standard deviation in a student-t distribution corresponding to the most probable run length at each time step (in medium timescale) is a byproduct of

Table 3.3: Traffic parameters and resource demands of simulated fBm traffic

Group	Simulated				Estimated (Benchmark)				Estimated (Proposed)				
	λ	σ	H	$\mathsf{R}_{0.01}$	$\bar{\lambda}$	$\bar{\sigma}$	$\bar{H}$	$\mathsf{R}_{0.01}$	$\bar{\lambda}$	$\bar{\sigma}$	$\bar{H}$	$\mathsf{R}_{0.01}$	$\mathtt{E}$
1	800	200	0.7	1429.6	797.9	216.3	0.698	1490.9	795.4	218.8	0.693	1501.4	0.1507
2	800	200	0.8	1403.4	804.4	191.4	0.796	1378.2	798.6	198.6	0.793	1397.8	0.1373
3	800	200	0.9	1422.5	801.3	182.4	0.894	1363.1	798.8	210.6	0.893	1453.2	0.1246
4	900	300	0.7	1895.7	898.5	325.6	0.697	1997.1	895.8	327.3	0.693	2008.3	0.1489
5	900	300	0.8	1836.4	896.9	287.5	0.796	1791.4	895.3	296.4	0.792	1822.0	0.1380
6	900	300	0.9	1848.7	899.6	271.8	0.895	1752.1	898.9	310.8	0.891	1879.0	0.1218

change point detection. It is observed that both statistics are stable between the detected change points. We see that both conditions in (3.16) and (3.17) are satisfied for the two detected change points, i.e., the most probable run length drops by more than Υ_l, and the change in mean plus standard deviation is sufficiently large. We also observe that the change points are detected after the occurrence of statistical changes, which verifies the effectiveness of the *look-back* traffic parameter learning.

To evaluate the accuracy of the proposed traffic parameter learning method, we simulate six groups of fBm traffic traces in discrete time with different traffic parameters and resource demands, as given in Table 3.3. The simulated fBm traffic is generated following a wavelet-based algorithm [83]. The length of a time unit is not specified. The resource demand to satisfy the QoS requirement $\mathbb{P}(\mathsf{d} > 0.01 \text{ time units}) \leq 0.01$ is denoted by $\mathsf{R}_{0.01}$. For each group of fBm traffic, 200 sample paths are generated, with 1000 traffic samples in each sample path. The traffic parameters of each sample path are estimated by the first 800 traffic samples using both the proposed learning method and a classical benchmark method. In the benchmark method, the mean and variance are estimated by the sample mean and variance, and the Hurst parameter is estimated separately using a wavelet-based approach [83]. In the proposed learning method, the three parameters are learned together, reaching a compromise among them to maximize the log-marginal likelihood function given in (3.23). The results of both methods are given in Table 3.3. It is observed that the mean and Hurst parameter given by both methods are close to the simulated parameters, but the standard deviation is not as accurate. However, the level of underestimation given by the learning method is much lower than that given by the benchmark method. The accuracy of traffic parameter learning is also evaluated by the average prediction error, $\mathtt{E}$, for the

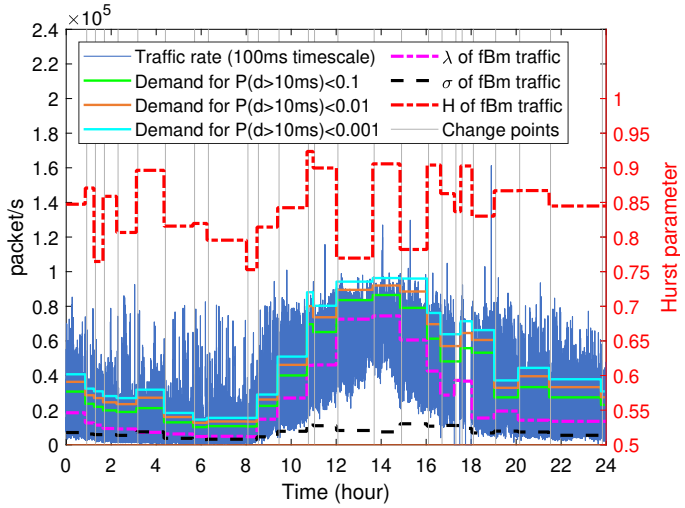


Figure 3.6: Learned traffic parameters and predicted resource demands for daily traffic.

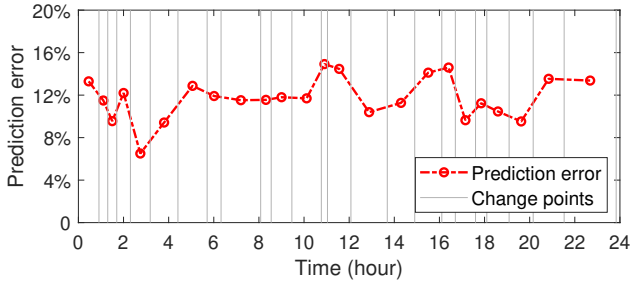
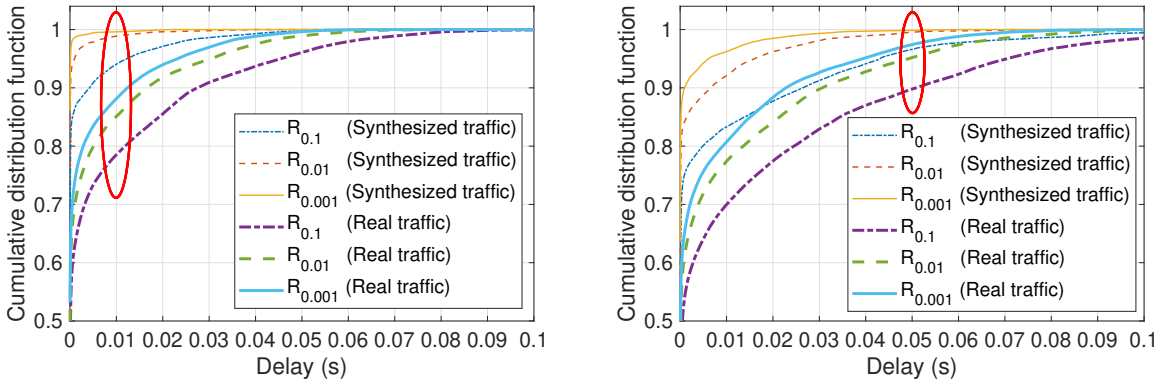


Figure 3.7: Evaluation of traffic parameter learning accuracy for daily traffic.

last 200 traffic samples in each sample path, as given in Table 3.3.

Fig. 3.6 shows results of the proposed change-point-driven traffic parameter learning and resource demand prediction scheme for a real-world daily traffic trace. The detected change points identify different stationary traffic segments. For each stationary traffic segment, the three learned fBm traffic parameters $\{\lambda(k), \sigma(k), H(k)\}$ are plotted. We observe that the Hurst parameter is within $[0.5, 1)$, indicating self-similarity and LRD of the traffic. The predicted resource demands for the identified stationary traffic segments are given, for QoS requirements $\mathbb{P}(\mathsf{d} > 10ms) \leq \varepsilon$ with $\varepsilon = 0.1$, 0.01, and 0.001. As expected, the resource demand increases when ε decreases. The average prediction error evaluated by $t_0 = 1000$ traffic samples in each identified stationary traffic segment is plotted in Fig. 3.7. It can



(a) $D = 10$ ms (b) $D = 50$ ms

Figure 3.8: Distribution of VNF packet processing delay for both the synthesized traffic and the real traffic.

be seen that the average prediction errors for the real-world traffic trace is comparable to that of the simulated fBm traffic traces given in Table 3.3.

To evaluate QoS performance of the proposed resource demand prediction scheme, we conduct packet-level simulations using the python Simpy package, to gather sufficient packet delay information for a smooth characterization of the VNF packet processing delay distribution, with a $60s$-long stationary traffic segment from the real-world traffic trace as the VNF traffic input. Different amount of resources are allocated to the VNF according to the predicted resource demands for different QoS requirements. For simplicity, we use R_ε to represent the predicted resource demand for a probabilistic delay guarantee, i.e., $\mathbb{P}(\mathsf{d} > \mathsf{D}) \leq \varepsilon$. Since traffic parameter learning is performed in 0.1 s timescale, traffic burstiness in time granularities smaller than 0.1 s cannot be captured. Hence, we use both the real packet arrival trace and a less-bursty synthesized packet arrival trace for QoS evaluation. In the synthesized packet arrival trace, the numbers of packet arrivals in 0.1 s timescale are the same as the real packet arrival trace, but the packet inter-arrival time within each 0.1 s time interval follows an exponential distribution. Fig. 3.8 shows the distribution of VNF packet processing delay for both traffic traces. Two groups of delay requirements with different delay bounds, i.e., $\mathsf{D} = 10$ ms and $\mathsf{D} = 50$ ms, are used for QoS evaluation. In each group, ε is set as 0.1, 0.01, and 0.001. For the same QoS requirement, the amount of resources allocated for both traffic are the same. However, the delay performance of the synthesized traffic is better than that of the real traffic, due to

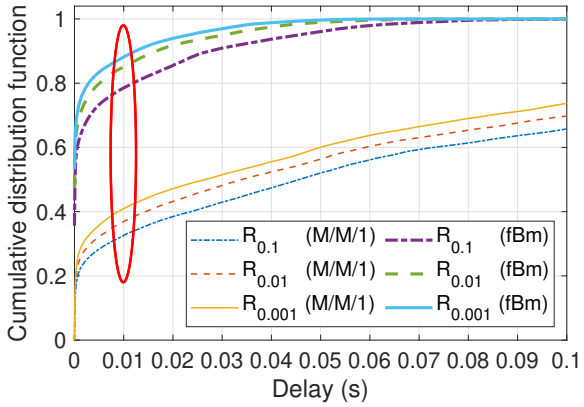
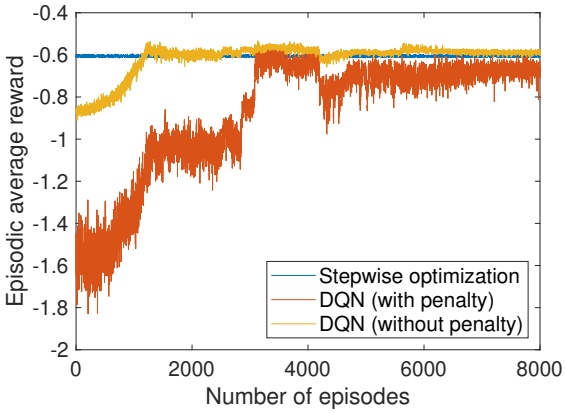


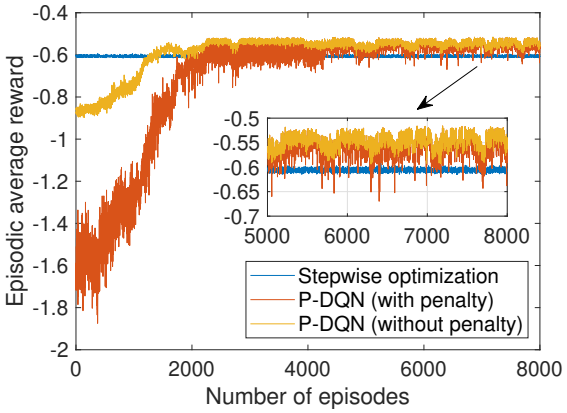
Figure 3.9: QoS performance comparison between the fBm model and M/M/1 model based resource demand prediction schemes.

less traffic burstiness in time granularities smaller than 0.1 s. For the synthesized traffic, the delay violation probability is within the corresponding upper limits. For the real traffic, the delay violation probability occasionally exceeds the required upper limit, especially for the stringent QoS requirements such as $\mathbb{P}(\mathsf{d} > 10ms) \leq 0.001$, due to traffic burstiness in time granularities below 0.1 s.

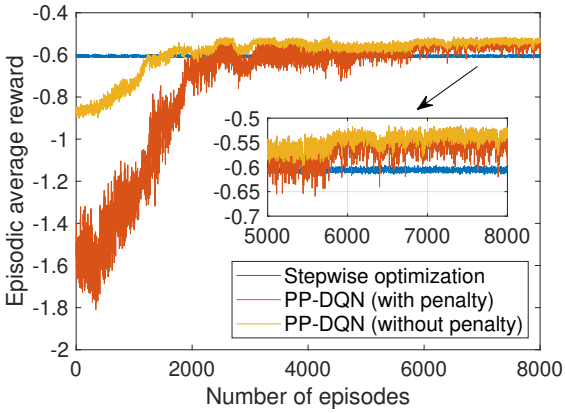
In addition, we compare the QoS performance between the proposed fBm model based resource demand prediction scheme and a benchmark M/M/1 model based counterpart. Both methods use the learned traffic parameters for resource demand prediction. In the proposed scheme, the resource demand is predicted from all three learned traffic parameters, i.e., $\{\lambda, \sigma, H\}$, based on the fBm resource provisioning model given by (3.31). In the benchmark scheme, the resource demand is predicted from the first learned traffic parameter, i.e., λ, based on an M/M/1 resource provisioning model. For an M/M/1 queue with arrival rate λ (in packet/s) and service rate R (in packet/s), the delay violation probability is $\mathbb{P}(\mathsf{d} > \mathsf{D}) = e^{-(\mathsf{R}-\lambda)\mathsf{D}}$ [84]. Hence, the minimum amount of resources (in packet/s) to guarantee the QoS requirement $\mathbb{P}(\mathsf{d} > \mathsf{D}) \leq \varepsilon$ is $\mathsf{R}_{min} = \lambda - \frac{\log \varepsilon}{\mathsf{D}}$. Fig. 3.9 shows the VNF packet delay distribution with the real packet arrivals and with different amount of resources allocated to the VNF, based on predicted resource demands given by the two models. A gap is observed between the delay performance of the two models. The proposed model, with the ability to capture the bursty nature of traffic, gives a better estimation of resource demands.



(a) D$\mathcal{Q}$N



(b) P-D$\mathcal{Q}$N



(c) PP-D$\mathcal{Q}$N

Figure 3.10: Episodic average reward versus the episode number for the three deep $\mathcal{Q}$-learning algorithms.

The performance of the proposed deep $\mathcal{Q}$-learning algorithm with penalty-aware prioritized experience replay (PP-D$\mathcal{Q}$N) is compared with two benchmark algorithms, i.e., deep $\mathcal{Q}$ learning with uniformly sampled experience replay (D$\mathcal{Q}$N), and deep $\mathcal{Q}$-learning with prioritized experience relay (P-D$\mathcal{Q}$N). All three deep $\mathcal{Q}$-learning algorithms are compared with a common benchmark, i.e., stepwise optimization. The comparison is performed using traffic set 1 in Table 3.1. Fig. 3.10 shows the evolution of episodic average reward with respect to the number of episodes during the learning process, using the three deep $\mathcal{Q}$-learning algorithms. Both the full reward including penalty and the partial reward without penalty are plotted, with a gap indicating the penalty. It is observed that D$\mathcal{Q}$N converges to a poor solution which is worse than the stepwise optimization benchmark in terms of episodic average reward. The penalty is high, inferring that the D$\mathcal{Q}$N does not learn a solution to minimize the resource overloading penalty in the long run. Both the P-D$\mathcal{Q}$N and PP-D$\mathcal{Q}$N algorithms take advantages of the prioritized experience replay for convergence to solutions that outperform the stepwise optimization benchmark in most time after convergence. It demonstrates that both P-D$\mathcal{Q}$N and PP-D$\mathcal{Q}$N after convergence can capture the daily and weekly traffic patterns (in both change points and resource demands) and background resource loading patterns at the candidate NFV nodes, and make intelligent VNF migration decisions accordingly. In contrast, when a VNF migration is required, the stepwise optimization benchmark favors VNF migration to a lightly loaded NFV node in the current decision epoch, which can be heavily loaded in the following decision epochs. As illustrated in Fig. 3.10(b) and Fig. 3.10(c), the proposed PP-D$\mathcal{Q}$N achieves slightly more gain in terms of penalty suppression compared with P-D$\mathcal{Q}$N. The episodic average rewards (with penalty) of P-D$\mathcal{Q}$N and PP-D$\mathcal{Q}$N after convergence are -0.5502 and -0.5408 respectively.

Fig. 3.11 shows that the training loss of PP-D$\mathcal{Q}$N as defined in (3.43) converges faster to a smaller value. We also examined the learning curve of PP-D$\mathcal{Q}$N with $\omega^{(\mathtt{TD})} = 0.5$, which gives an average training loss of 0.0155 after convergence, demonstrating the benefit of reducing weight $\omega^{(\mathtt{TD})}$ on further loss reduction. However, the benefit on additional reward improvement is not significant. To evaluate generalization of the proposed algorithm to similar traffic sets with different randomness, we compare the average training loss of both P-D$\mathcal{Q}$N and PP-D$\mathcal{Q}$N after convergence in Fig. 3.12, using four traffic sets with different randomness levels in change points and resource demands, as in Table 3.1. With more randomness especially in resource demand, the average training loss of both P-D$\mathcal{Q}$N and PP-D$\mathcal{Q}$N increases. However, the PP-D$\mathcal{Q}$N outperforms P-D$\mathcal{Q}$N for all the traffic sets.

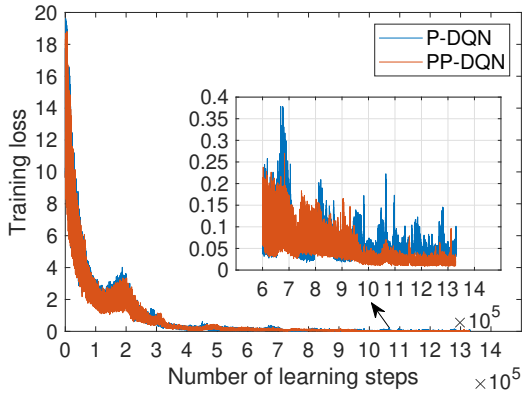


Figure 3.11: Training loss of the evaluation $\mathcal{Q}$ networks.

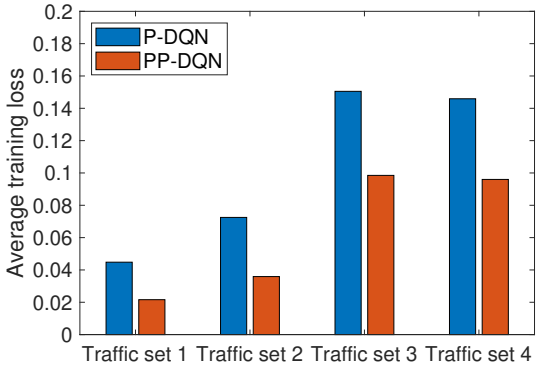


Figure 3.12: Average training loss after convergence.

3.5 Summary

In this chapter, we study a dynamic VNF scaling problem in a local network scenario with several candidate NFV nodes, to guarantee probabilistic delay performance in the presence of real-world traffic with nonstationary characteristics. A change-point-driven traffic parameter learning and resource demand prediction scheme is proposed, based on which dynamic VNF migration decisions are made at variable-length decision epochs using a penalty-aware deep $\mathcal{Q}$-learning algorithm. An fBm traffic model is employed for each identified stationary traffic segment, based on properties of Gaussianity and self-similarity of the real-world traffic. The proposed traffic parameter learning method achieves a better accuracy than a benchmark method for the simulated fBm traffic, benefiting from a compromise among the three traffic parameters. The traffic parameter learning accuracy is also demonstrated by the average prediction error with the trained GPR models. For the proposed resource demand prediction scheme, packet-level simulations show occasional QoS violation for a bursty real-world packet arrival trace especially for the stringent QoS requirements, and QoS satisfaction for a synthesized packet arrival trace with less traffic burstiness. The proposed penalty-aware deep $\mathcal{Q}$-learning algorithm achieves performance gains in terms of both training loss reduction and episodic average reward maximization.

Chapter 4

Delay-Aware VNF Scheduling For Network Utility Maximization

In this chapter, a delay-aware VNF scheduling problem is studied in the presence of small-timescale traffic dynamics, to achieve network utility maximization with minimum throughput guarantee for each deadline-constrained service, while using a realistic time quantum in the $100\mu s$ to ms granularity for CPU resource scheduling. Based on the Lyapunov optimization technique [85], an online distributed VNF scheduling algorithm is derived, which greedily schedules a VNF at each NFV node based on a weight incorporating the backpressure-based weighted differential backlogs, the throughput performance, and the packet delay. Simulation results demonstrate an $\left[\mathcal{O}(\frac{1}{\vartheta}), \mathcal{O}(\vartheta)\right]$ utility-backlog trade-off with utility importance parameter ϑ. The proposed VNF scheduling algorithm using a realistic time quantum achieves a comparable performance with a generalized processor sharing (GPS) scheme under the assumption of infinitely divisible resources. We also evaluate the impact of packet rushing (i.e., packet processing by multiple consecutive VNFs in one time slot) on VNF scheduling performance, based on a packet rushing analysis.

4.1 System Model

4.1.1 Services

Consider multiple services in a time-slotted system where time is partitioned into equal-length time slots indexed by τ. The time slot length is T in second. Each service is in the form of VNF chain, originating from an ingress edge switch and traversing through a sequence of VNFs towards an egress edge switch. Let $\mathcal{R}$ denote the set of services. Let H_r be the number of VNFs in service $r \in \mathcal{R}$, and let $\mathcal{H}_r = \{1, \cdots, H_r\}$ be a set containing the index of VNFs of service $r \in \mathcal{R}$. Denote the h-th ($h \in \mathcal{H}_r$) VNF in SFC $r \in \mathcal{R}$ as $V_h^{(r)}$. Under the assumption of same virtualization platform at the NFV nodes, let $P_h^{(r)}$ denote the processing density of VNF $V_h^{(r)}$ at any NFV nodes in cycle/packet, which is the CPU resource demand (in cycle/s) of VNF $V_h^{(r)}$ for one packet/s of processing rate [62,64]. We assume that the exogenous packet arrivals for each service occur only at the ingress edge switch. The arrival process is stationary and ergodic with mean rate $\bar{a}_r$ (in packet per time slot) for service $r \in \mathcal{R}$. Let $A^{(r)}(\tau)$ denote the number of packets that arrive at the τ-th time slot for service $r \in \mathcal{R}$, which is highly dynamic and unpredictable. Assume that $A^{(r)}(\tau)$ is upper bounded by a finite maximum value $A_{max}^{(r)}$ for service $r \in \mathcal{R}$. The QoS requirement of service $r \in \mathcal{R}$ is represented by two parameters, M_r and ε_r, where M_r is the E2E deadline (in time slots) for each packet of service $r \in \mathcal{R}$, and ε_r specifies the maximum packet dropping ratio of service $r \in \mathcal{R}$. For a deadline-constrained service, a packet becomes useless once the E2E delay is violated and should be dropped. For service $r \in \mathcal{R}$, only the timely delivered packets to the egress edge switch within E2E deadline M_r are counted in the throughput, denoted by $\bar{f}_r$ (in packet per time slot), which should be at least $\bar{a}_r(1 - \varepsilon_r)$ to satisfy the QoS requirement.

4.1.2 Network Model

We consider a core network with a set $\mathcal{N}$ of NFV nodes interconnected by virtual links. The CPU processing resource budget at NFV node $n \in \mathcal{N}$ is C_n in cycle per time slot. Assume that there are sufficient transmission resources in the network for virtual link provisioning to enable communications among the NFV nodes. The services in set $\mathcal{R}$ are embedded in the network with VNF placement at NFV nodes and traffic routing over virtual links. Here, we consider fixed VNF placement and traffic routing. Let x_n^{rh} be a binary parameter,

with $\mathrm{x}_n^{rh} = 1$ if VNF $V_h^{(r)}$ is placed at NFV node $n \in \mathcal{N}$, and $\mathrm{x}_n^{rh} = 0$ otherwise. Let $\mathcal{V}_n = \{(r,h) | r \in \mathcal{R}, h \in \mathcal{H}_r, \mathrm{x}_n^{rh} = 1\}$ be a set containing the index of all VNFs placed at NFV node $n \in \mathcal{N}$, with $(r,h) \in \mathcal{V}_n$ denoting VNF $V_h^{(r)}$. Let $z_h^{(r)}(\tau)$ be a binary VNF scheduling decision variable, with $z_h^{(r)}(\tau) = 1$ if VNF $V_h^{(r)}$ is scheduled for packet processing at the corresponding NFV node during time slot τ and $z_h^{(r)}(\tau) = 0$ otherwise.

4.1.3 Queueing Model

Assumptions

Under the assumption that the virtual link delay is negligible, all packets processed by VNF $V_h^{(r)}$ during time slot τ can arrive at the downstream VNF $V_{h+1}^{(r)}$ for $h \in \mathcal{H}_r \backslash \{H_r\}$ or at the egress edge switch for $h = H_r$ before the beginning of time slot $\tau + 1$. Existing studies usually assume that a packet can be processed by at most one VNF in a chain during one time slot (i.e., no packet rushing) [62]. We first develop a VNF scheduling algorithm relying on the assumption of no packet rushing, and then propose a modified VNF scheduling algorithm with packet rushing.

Physical packet processing queues

NFV node $n \in \mathcal{N}$ maintains a separate packet processing queue for each VNF in set $\mathcal{V}_n$. Let $q_h^{(r)}(\tau)$ denote the number of packets in the queue associated with VNF $V_h^{(r)}$ at the beginning of time slot τ. Let $S_h^{(r)}(\tau)$ and $D_h^{(r)}(\tau)$ be the number of packets processed and dropped from the queue of VNF $V_h^{(r)}$ during time slot τ, respectively. Then, the queue length of VNF $V_h^{(r)}$ at the beginning of time slot $\tau + 1$, i.e., $q_h^{(r)}(\tau + 1)$, is updated as

$$q_h^{(r)}(\tau+1) = \left[q_h^{(r)}(\tau) - S_h^{(r)}(\tau) - D_h^{(r)}(\tau)\right]^+ + S_{h-1}^{(r)}(\tau)\mathbb{1}\{h > 1\} + A^{(r)}(\tau)\mathbb{1}\{h = 1\}, \quad \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r \tag{4.1}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, equal to 1 only if the condition inside the bracket is true and 0 otherwise.

Delay-aware virtual packet processing queues

For service $r \in \mathcal{R}$, a new packet admitted to the first VNF $V_1^{(r)}$ in the chain has a zero packet delay. The delay (in number of time slots) of a packet is increased by 1, for each time slot. If a packet of service $r \in \mathcal{R}$ cannot be successfully delivered to the egress edge switch before the E2E deadline of M_r, it is expired and dropped. Under the assumption of no packet rushing, the delay of any packets at VNF $V_h^{(r)}$ is larger than or equal to $h-1$ time slots, i.e., the residual packet lifetime does not exceed $M_r - h + 1$ time slots at VNF $V_h^{(r)}$. Also, since there are $H_r - h$ downstream VNFs after VNF $V_h^{(r)}$, a packet with a residual lifetime less than $H_r - h + 1$ time slots at VNF $V_h^{(r)}$ cannot successfully reach the egress edge switch before expiry even if the packet does not wait at all downstream VNFs. To avoid resource inefficiency for processing such packets at the downstream VNFs, a packet with a residual lifetime of $H_r - h + 1$ time slots during a certain time slot at VNF $V_h^{(r)}$ is dropped if it is not processed before the end of the time slot. Hence, all the existing packets at VNF $V_h^{(r)}$ during any time slot have a residual lifetime $m \in \mathcal{M}_h^{(r)}$, where $\mathcal{M}_h^{(r)}$ is a set given by

$$\mathcal{M}_h^{(r)} = \{H_r - h + 1, \cdots, M_r - h + 1\}, \quad \forall r \in \mathcal{R},\ h \in \mathcal{H}_r. \tag{4.2}$$

At VNF $V_h^{(r)}$, all the packets with residual lifetime $m = H_r - h + 1$ are referred to as urgent packets, and other packets with residual lifetime $m \in \mathcal{M}_h^{(r)} \backslash \{H_r - h + 1\}$ are non-urgent packets.

For VNF $V_h^{(r)}$ of service $r \in \mathcal{R}$, there are $|\mathcal{M}_h^{(r)}|$ delay-aware virtual packet processing queues, denoted by $\{Q_{h,m}^{(r)}, \forall m \in \mathcal{M}_h^{(r)}\}$, each of which corresponds to a residual packet lifetime of $m \in \mathcal{M}_h^{(r)}$. A packet is virtually associated with a virtual packet processing queue according to its residual lifetime. For a service with 3 VNFs in the chain and a packet E2E deadline of 6 in time slot, each VNF is associated with $|\mathcal{M}_h^{(r)}| = 4$ virtual packet processing queues, as illustrated in Fig. 4.1. Let $Q_{h,m}^{(r)}(\tau)$ be the length of queue $Q_{h,m}^{(r)}$ (i.e., the number of packets with residual lifetime $m \in \mathcal{M}_h^{(r)}$ at VNF $V_h^{(r)}$) at the beginning of time slot τ, with $q_h^{(r)}(\tau) = \sum_{m \in \mathcal{M}_h^{(r)}} Q_{h,m}^{(r)}(\tau)$. Let $S_{h,m}^{(r)}(\tau)$ be the number of packets with residual lifetime $m \in \mathcal{M}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ, with $S_h^{(r)}(\tau) = \sum_{m \in \mathcal{M}_h^{(r)}} S_{h,m}^{(r)}(\tau)$. Then, all the packets in queue $Q_{h,m}^{(r)}$ ($m \in \mathcal{M}_h^{(r)} \backslash \{H_r - h + 1\}$) during time slot τ have a residual lifetime of $m-1$ at the beginning of time slot $\tau+1$, which is equivalent to 1) transferring $S_{h,m}^{(r)}(\tau)$ packets from queue $Q_{h,m}^{(r)}$ to queue $Q_{h+1,m-1}^{(r)}$ at the

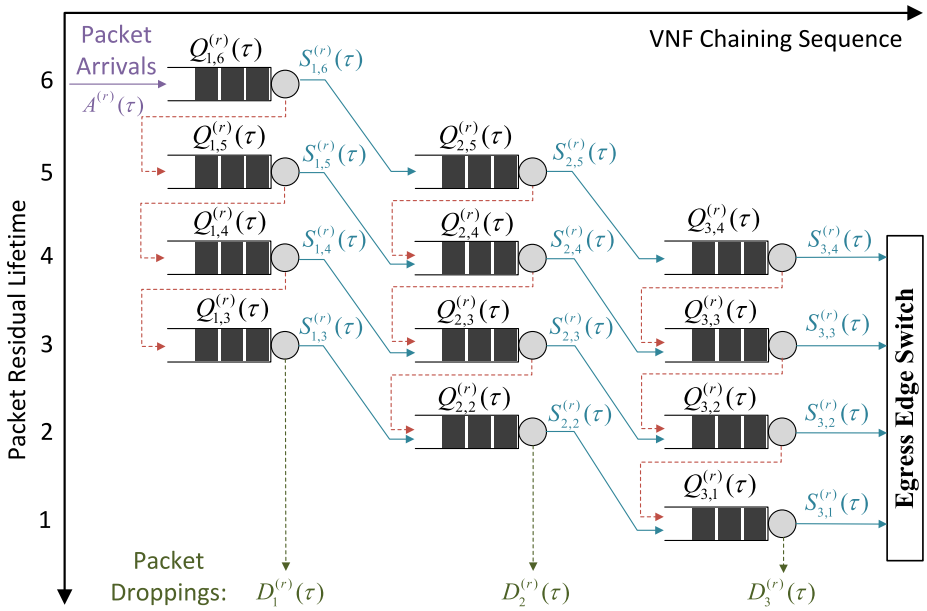


Figure 4.1: An illustration of delay-aware virtual packet processing queueing model for service r with $H_r = 3$ and $M_r = 6$.

next VNF (if $h < H_r$) and 2) transferring the remaining $\left[Q^{(r)}_{h,m}(\tau) - S^{(r)}_{h,m}(\tau)\right]^+$ packets from queue $Q^{(r)}_{h,m}$ to queue $Q^{(r)}_{h,m-1}$ at the same VNF during time slot τ. Taking the first VNF in Fig. 4.1 as an example, the new arrived packets with a full residual lifetime of 6 are all associated with virtual packet processing queue $Q^{(r)}_{1,6}$. After one time slot, the processed packets at queue $Q^{(r)}_{1,6}$ are transferred to queue $Q^{(r)}_{2,5}$ at the second VNF, with the residual lifetime decreased by 1. The remaining unprocessed packets are all transferred to a lower-layer queue $Q^{(r)}_{1,5}$ at the first VNF corresponding to a packet residual lifetime minus 1. The number of packets dropped from VNF $V^{(r)}_h$ during time slot τ, i.e., $D^{(r)}_h(\tau)$, is equal to the number of unprocessed urgent packets at VNF $V^{(r)}_h$, represented by

$$D^{(r)}_h(\tau) = \left[Q^{(r)}_{h,H_r-h+1}(\tau) - S^{(r)}_{h,H_r-h+1}(\tau)\right]^+, \quad \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r. \tag{4.3}$$

Accordingly, the queueing dynamics of the delay-aware virtual packet processing queues of service $r \in \mathcal{R}$ are updated as

$$Q^{(r)}_{1,M_r}(\tau+1) = A^{(r)}(\tau), \qquad \forall r \in \mathcal{R}$$

$$
\begin{aligned}
&Q_{1,m}^{(r)}(\tau+1) = \left[Q_{1,m+1}^{(r)}(\tau) - S_{1,m+1}^{(r)}(\tau)\right]^+, && \forall r \in \mathcal{R}, \quad \forall m \in \mathcal{M}_1^{(r)} \backslash \{M_r\} \\
&Q_{h,M_r-h+1}^{(r)}(\tau+1) = S_{h-1,M_r-h+2}^{(r)}(\tau), && \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r \backslash \{1\} \\
&Q_{h,m}^{(r)}(\tau+1) = \left[Q_{h,m+1}^{(r)}(\tau) - S_{h,m+1}^{(r)}(\tau)\right]^+ + S_{h-1,m+1}^{(r)}(\tau), && \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r \backslash \{1\}, \\
& && \forall m \in \mathcal{M}_h^{(r)} \backslash \{M_r - h + 1\}
\end{aligned} \tag{4.4}
$$

which combines the evolution for both the physical packet processing queue lengths in (4.1) and the packet delay. The average E2E delay (in second) of the timely delivered packets of service $r \in \mathcal{R}$ to the egress edge switch, denoted by $\bar{d}_r$, is calculated as

$$
\bar{d}_r = \sum_{m \in \mathcal{M}_{H_r}^{(r)}} \left[(M_r - m + 1)\, T \frac{\lim_{\Gamma \to \infty} \frac{1}{\Gamma} \sum_{\tau=0}^{\Gamma-1} \mathbb{E}\left\{ S_{H_r,m}^{(r)}(\tau) \right\}}{\sum_{m \in \mathcal{M}_{H_r}^{(r)}} \lim_{\Gamma \to \infty} \frac{1}{\Gamma} \sum_{\tau=0}^{\Gamma-1} \mathbb{E}\left\{ S_{H_r,m}^{(r)}(\tau) \right\}} \right], \quad \forall r \in \mathcal{R} \tag{4.5}
$$

where T is the time slot length in second, Γ is the total number of time slots, and $\mathbb{E}$ denotes expectation over the randomness in packet arrivals, VNF scheduling, and packet processing.

4.2 Problem Formulation

To meet the QoS requirements of all the deadline-constrained services, we investigate a delay-aware VNF scheduling problem, to determine which VNF to schedule at each NFV node at each time slot τ, i.e., $\boldsymbol{z}(\tau) = \{z_h^{(r)}(\tau), \forall r, \forall h\}$, and how many packets of each delay to be processed from the VNFs at each time slot τ, i.e., $\boldsymbol{S}(\tau) = \{S_{h,m}^{(r)}(\tau), \forall r, \forall h, \forall m\}$, while maximizing a fairness-aware total network utility with throughput guarantee for each service.

The throughput of a deadline-constrained service is a timely throughput, since only the timely delivered packets to the egress edge switch within E2E deadline M_r are counted in the throughput, and other packets are dropped once the E2E delay is violated. Hence, the throughput of deadline-constrained service $r \in \mathcal{R}$, denoted by $\bar{f}_r$, is given by

$$
\bar{f}_r = \lim_{\Gamma \to \infty} \frac{1}{\Gamma} \sum_{\tau=0}^{\Gamma-1} \mathbb{E}\left\{ A^{(r)}(\tau) - \sum_{h \in \mathcal{H}_r} D_h^{(r)}(\tau) \right\}, \quad \forall r \in \mathcal{R} \tag{4.6}
$$

under the mean rate stable condition for all the physical packet processing queues, represented by [85]

$$\lim_{\Gamma\to\infty} \frac{\mathbb{E}\left\{\sum_{r\in\mathcal{R}}\sum_{h\in\mathcal{H}_r} q_h^{(r)}(\Gamma)\right\}}{\Gamma} = 0. \tag{4.7}$$

To satisfy the QoS requirement, the throughput of service $r \in \mathcal{R}$ should satisfy

$$\bar{f}_r \geq \bar{a}_r(1-\varepsilon_r), \quad \forall r \in \mathcal{R}. \tag{4.8}$$

At each NFV node, at most one VNF can be scheduled during time slot τ, represented by

$$\sum_{(r,h)\in\mathcal{V}_n} z_h^{(r)}(\tau) = 1, \quad \forall n \in \mathcal{N}. \tag{4.9}$$

Only packets of the scheduled VNF can be processed, and the total number of CPU cycles consumed for processing packets by the scheduled VNF during a time slot should not exceed the allocated processing resources at the corresponding NFV node, given by

$$0 \leq P_h^{(r)} \sum_{m\in\mathcal{M}_h^{(r)}} S_{h,m}^{(r)}(\tau) \leq z_h^{(r)}(\tau) \sum_{n\in\mathcal{N}} \mathrm{x}_n^{rh} C_n, \quad \forall r \in \mathcal{R},\ \forall h \in \mathcal{H}_r. \tag{4.10}$$

Moreover, the number of packets with residual lifetime $m \in \mathcal{M}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ should not exceed the corresponding virtual packet processing queue length, given by

$$0 \leq S_{h,m}^{(r)}(\tau) \leq Q_{h,m}^{(r)}(\tau), \quad \forall r \in \mathcal{R},\ \forall h \in \mathcal{H}_r,\ \forall m \in \mathcal{M}_h^{(r)}. \tag{4.11}$$

We consider a strictly increasing and concave utility function $\phi(\cdot)$ of throughput with proportional fairness among services in terms of processing resources, given by

$$\phi(\bar{f}_r) = \log\left(P_r \bar{f}_r\right) \tag{4.12}$$

where $P_r = \sum_{h\in\mathcal{H}_r} P_h^{(r)}$ is the aggregate processing density (in cycle/packet) of service $r \in \mathcal{R}$. In (4.12), the throughput in packet per time slot of service $r \in \mathcal{R}$ (i.e., $\bar{f}_r$) is weighted by P_r, to represent the total throughput in cycle per time slot at all VNFs of

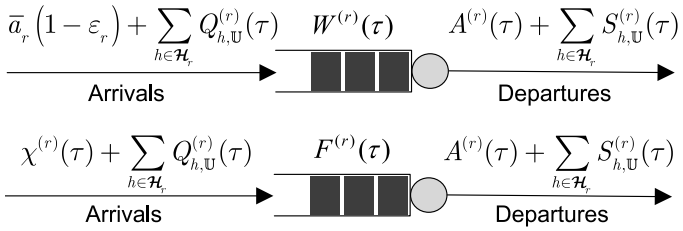


Figure 4.2: Virtual queues $W^{(r)}(\tau)$ and $F^{(r)}(\tau)$ for service $r \in \mathcal{R}$.

the service. Then, the VNF scheduling problem is formulated as a stochastic optimization problem, given by

$$\begin{aligned} \mathbf{P}_\infty: \max_{\{\boldsymbol{z}(\tau),\boldsymbol{S}(\tau),\forall\tau\}} \quad & \sum_{r\in\mathcal{R}} \phi(\bar{f}_r) \\ \text{s.t.} \quad & (4.7), (4.8), (4.9), (4.10), (4.11) \end{aligned} \tag{4.13}$$

where the objective function and constraints (4.7)-(4.8) involve infinite-horizon expectations, and constraints (4.9)-(4.11) are instantaneous constraints for each time slot.

Problem transformation using Lyapunov optimization

By introducing auxiliary decision variables $\boldsymbol{\chi}(\tau) = \{\chi^{(r)}(\tau) \in [0, A^{(r)}_{max}], \forall r \in \mathcal{R}\}$ for time slot τ, problem $\mathbf{P}_\infty$ is transformed to

$$\mathbf{P}'_\infty: \max_{\{\boldsymbol{z}(\tau),\boldsymbol{S}(\tau),\boldsymbol{\chi}(\tau),\forall\tau\}} \quad \lim_{\Gamma\to\infty} \frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1} \mathbb{E}\left\{\sum_{r\in\mathcal{R}} \phi\left(\chi^{(r)}(\tau)\right)\right\} \tag{4.14a}$$

$$\text{s.t.} \quad (4.7), (4.8), (4.9), (4.10), (4.11) \tag{4.14b}$$

$$\bar{\chi}_r \le \bar{f}_r, \quad \forall r \in \mathcal{R} \tag{4.14c}$$

where $\bar{\chi}_r = \lim_{\Gamma\to\infty} \frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1} \mathbb{E}\left\{\chi^{(r)}(\tau)\right\}$ is the infinite-horizon time average expectations of $\chi^{(r)}(\tau)$ for service $r \in \mathcal{R}$. Problem $\mathbf{P}'_\infty$ is equivalent to problem $\mathbf{P}_\infty$ [62, 85].

To handle the stochastic inequality constraints (4.8) and (4.14c), problem $\mathbf{P}'_\infty$ requires further transformation through introducing equivalent virtual queue stability constraints [85]. Introduce two virtual queues for service $r \in \mathcal{R}$, i.e., $W^{(r)}(\tau)$ and $F^{(r)}(\tau)$, as illustrated in Fig. 4.2, with queue length evolution equations given by

$$W^{(r)}(\tau+1) = \left[W^{(r)}(\tau) - A^{(r)}(\tau) - \sum_{h\in\mathcal{H}_r} S^{(r)}_{h,\mathbb{U}}(\tau)\right]^+ + \bar{a}_r\left(1-\varepsilon_r\right) + \sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau) \tag{4.15}$$

$$F^{(r)}(\tau+1)=\left[F^{(r)}(\tau)-A^{(r)}(\tau)-\sum_{h\in\mathcal{H}_r}S^{(r)}_{h,\mathbb{U}}(\tau)\right]^{+}+\chi^{(r)}(\tau)+\sum_{h\in\mathcal{H}_r}Q^{(r)}_{h,\mathbb{U}}(\tau) \tag{4.16}$$

where $Q^{(r)}_{h,\mathbb{U}}(\tau)$ and $S^{(r)}_{h,\mathbb{U}}(\tau)$ are the total and processed numbers of urgent packets in the queue of VNF $V^{(r)}_h$ during time slot τ respectively. Without packet rushing, we have $Q^{(r)}_{h,\mathbb{U}}(\tau)=Q^{(r)}_{h,H_r-h+1}(\tau)$ and $S^{(r)}_{h,\mathbb{U}}(\tau)=S^{(r)}_{h,H_r-h+1}(\tau)$. Specifically, from (4.15) we obtain $\frac{W^{(r)}(\Gamma)-W^{(r)}(0)}{\Gamma}+\frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1}A^{(r)}(\tau)\geq\bar{a}_r\left(1-\varepsilon_r\right)+\frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1}\sum_{h\in\mathcal{H}_r}D^{(r)}_h(\tau)$ where $D^{(r)}_h(\tau)=Q^{(r)}_{h,\mathbb{U}}(\tau)-S^{(r)}_{h,\mathbb{U}}(\tau)$. Taking expectations of both sides with $\Gamma\to\infty$ and using $W^{(r)}(0)=0$, we obtain $\lim_{\Gamma\to\infty}\frac{\mathbb{E}\{W^{(r)}(\Gamma)\}}{\Gamma}+\bar{f}_r\geq\bar{a}_r\left(1-\varepsilon_r\right)$. Then, the mean rate stability of virtual queue $W^{(r)}(\tau)$, represented by $\lim_{\Gamma\to\infty}\frac{\mathbb{E}\{W^{(r)}(\Gamma)\}}{\Gamma}=0$, guarantees the stochastic constraint of $\bar{f}_r\geq\bar{a}_r\left(1-\varepsilon_r\right)$ for service $r\in\mathcal{R}$. Similarly, the mean rate stability of virtual queue $F^{(r)}(\tau)$ enforces the constraint of $\bar{\chi}_r\leq\bar{f}_r$ for service $r\in\mathcal{R}$. Then, problem $\mathbf{P}'_\infty$ is further transformed to

$$\mathbf{P}''_\infty:\ \max_{\{\boldsymbol{z}(\tau),\boldsymbol{S}(\tau),\boldsymbol{\chi}(\tau),\forall\tau\}}\quad\lim_{\Gamma\to\infty}\frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1}\mathbb{E}\left\{\sum_{r\in\mathcal{R}}\phi\left(\chi^{(r)}(\tau)\right)\right\} \tag{4.17a}$$

$$\text{s.t.}\quad (4.7),(4.9),(4.10),(4.11) \tag{4.17b}$$

$$\lim_{\Gamma\to\infty}\frac{\mathbb{E}\{W^{(r)}(\Gamma)\}}{\Gamma}=\lim_{\Gamma\to\infty}\frac{\mathbb{E}\{F^{(r)}(\Gamma)\}}{\Gamma}=0,\quad\forall r\in\mathcal{R}. \tag{4.17c}$$

Let $\boldsymbol{q}(\tau)=\{q^{(r)}_h(\tau),\forall r,\forall h\}$, $\boldsymbol{F}(\tau)=\{F^{(r)}(\tau),\forall r\}$ and $\boldsymbol{W}(\tau)=\{W^{(r)}(\tau),\forall r\}$ be the physical and virtual queue vectors at time slot τ. Let $\boldsymbol{\Theta}(\tau)=[\boldsymbol{q}(\tau),\boldsymbol{W}(\tau),\boldsymbol{F}(\tau)]$ be the combined queue vector at time slot τ. Without loss of generality, assume that all queue buffers are infinite and all queues are initially empty at $\tau=0$. Define Lyapunov function $L\left(\boldsymbol{\Theta}(\tau)\right)$ as a scalar metric of congestion level in the queueing system, given by

$$L\left(\boldsymbol{\Theta}(\tau)\right)=\frac{1}{2}\left\{\sum_{r\in\mathcal{R}}\sum_{h\in\mathcal{H}_r}\left[P^{(r)}_h q^{(r)}_h(\tau)\right]^2+\sum_{r\in\mathcal{R}}\left[P_r F^{(r)}(\tau)\right]^2+\sum_{r\in\mathcal{R}}\left[\varphi\frac{W^{(r)}(\tau)}{\bar{a}_r}\right]^2\right\}. \tag{4.18}$$

Note that the delay-aware virtual packet processing queue lengths $\{Q^{(r)}_{h,m}(\tau),\forall r,\forall h,\forall m\}$ are not directly included in the Lyapunov function, since the stability of physical packet processing queues infer the stability of delay-aware virtual packet processing queues. However, the urgent packet queue lengths are implicitly incorporated via virtual queues $W^{(r)}(\tau)$ and $F^{(r)}(\tau)$ according to (4.15) and (4.16). In (4.18), $q^{(r)}_h(\tau)$ is weighted by the processing density $P^{(r)}_h$ of VNF $V^{(r)}_h$, to represent a processing queue backlog in the number

of required CPU cycles. The virtual queue length $F^{(r)}(\tau)$ is weighted by the aggregate processing density P_r of service $r \in \mathcal{R}$, since the throughput of different services are weighted by the individual aggregate processing densities in the total network utility according to (4.12). The virtual queue length $W^{(r)}(\tau)$ is normalized by the average number of packet arrivals in each time slot, i.e., $\bar{a}_r$, and then rescaled using a weight $\varphi = [\max_{r\in\mathcal{R}} \bar{a}_r] \left[\max_{r\in\mathcal{R}, h\in\mathcal{H}_r} P_h^{(r)}\right]$, to place equal importance on the throughput guarantee for each service regardless of their packet arrival rates and processing densities. A simplified Lyapunov function $L\left(\mathbf{\Theta}(\tau)\right) = \frac{1}{2}\left\{\sum_{r\in\mathcal{R}} \sum_{h\in\mathcal{H}_r} \left[P_h^{(r)} q_h^{(r)}(\tau)\right]^2\right\}$ which is unaware of the virtual queue congestion levels corresponds to a classical backpressure algorithm adapted for processing resources. To keep the physical and virtual queues stable by persistently pushing the Lyapunov function $L\left(\mathbf{\Theta}(\tau)\right)$ towards a lower congestion level, a one-step conditional Lyapunov drift $\Delta\left(\mathbf{\Theta}(\tau)\right)$ is introduced, given by

$$\Delta\left(\mathbf{\Theta}(\tau)\right) = \mathbb{E}\left\{L\left(\mathbf{\Theta}(\tau+1)\right) - L\left(\mathbf{\Theta}(\tau)\right) | \mathbf{\Theta}(\tau)\right\}. \tag{4.19}$$

Based on the Lyapunov optimization theory, we can decouple the decision variables of problem $\mathbf{P}''_{\infty}$ over time slots and achieve asymptotically optimal total utility with queue stability, by solving an instantaneous problem for each time slot [85–88]. At time slot τ, the upper bound of a conditional Lyapunov drift-plus-penalty (or drift-minus-utility) term defined as $\Delta\left(\mathbf{\Theta}(\tau)\right) - \vartheta\mathbb{E}\left\{\sum_{r\in\mathcal{R}} \phi(\chi^{(r)}(\tau)) | \mathbf{\Theta}(\tau)\right\}$ is minimized, subject to all the instantaneous constraints (4.9)-(4.11). Here, ϑ (> 0) is a utility importance parameter that balances the importance between utility maximization and queue backlog reduction. The upper bound of the conditional Lyapunov drift-plus-penalty is given in Lemma 1.

Lemma 1. *Regardless of the randomness in packet arrivals and VNF scheduling decisions, the conditional Lyapunov drift-plus-penalty at time slot τ, has an upper bound, given by*

$$\begin{aligned}
&\Delta\left(\mathbf{\Theta}(\tau)\right) - \vartheta\mathbb{E}\left\{\sum_{r\in\mathcal{R}} \phi\left(\chi^{(r)}(\tau)\right) | \mathbf{\Theta}(\tau)\right\} \\
&\leq \sum_{r\in\mathcal{R}} \mathsf{B}_r + \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\left[\left(P_1^{(r)}\right)^2 q_1^{(r)}(\tau) - \left(\frac{\varphi}{\bar{a}_r}\right)^2 W^{(r)}(\tau) - \left(P_r\right)^2 F^{(r)}(\tau)\right] A^{(r)}(\tau) | \mathbf{\Theta}(\tau)\right\} \\
&\quad + \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\frac{\varphi^2\left(1-\varepsilon_r\right)}{\bar{a}_r} W^{(r)}(\tau) + \left[\left(\frac{\varphi}{\bar{a}_r}\right)^2 W^{(r)}(\tau) + \left(P_r\right)^2 F^{(r)}(\tau)\right] \sum_{h\in\mathcal{H}_r} Q_{h,\mathbb{U}}^{(r)}(\tau) | \mathbf{\Theta}(\tau)\right\} \\
&\quad - \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\Phi_1^{(r)}(\tau) | \mathbf{\Theta}(\tau)\right\} - \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\Phi_2^{(r)}(\tau) | \mathbf{\Theta}(\tau)\right\}
\end{aligned} \tag{4.20}$$

where B_r *is a constant given in Appendix C, and* $\Phi_1^{(r)}(\tau)$, $\Phi_2^{(r)}(\tau)$ *are given by*

$$\Phi_1^{(r)}(\tau) = \vartheta \cdot \phi\left(\chi^{(r)}(\tau)\right) - (P_r)^2 \left[F^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} Q_{h,\mathbb{U}}^{(r)}(\tau)\right] \chi^{(r)}(\tau), \quad \forall r \in \mathcal{R} \tag{4.21}$$

$$\begin{aligned}\Phi_2^{(r)}(\tau) = &\sum_{h\in\mathcal{H}_r} \left(P_h^{(r)}\right)^2 q_h^{(r)}(\tau) \left[S_h^{(r)}(\tau) - S_{h-1}^{(r)}(\tau)\mathbb{1}\{h>1\}\right] \\ &+ \sum_{h\in\mathcal{H}_r} \left[\left(\frac{\varphi}{\bar{a}_r}\right)^2 W^{(r)}(\tau) + (P_r)^2 F^{(r)}(\tau) - \left(P_h^{(r)}\right)^2 q_h^{(r)}(\tau)\right] S_{h,\mathbb{U}}^{(r)}(\tau), \ \forall r \in \mathcal{R}.\end{aligned} \tag{4.22}$$

Proof: See Appendix C.

Since only the last two terms of the upper bound in (4.20) are related to the decision variables, we can formulate an instantaneous problem at time slot τ as

$$\begin{aligned}\mathbf{P}_\tau : \max_{\boldsymbol{z}(\tau),\boldsymbol{S}(\tau),\boldsymbol{\chi}(\tau)} \quad & \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\Phi_1^{(r)}(\tau)|\boldsymbol{\Theta}(\tau)\right\} + \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\Phi_2^{(r)}(\tau)|\boldsymbol{\Theta}(\tau)\right\} \\ \text{s.t.} \quad & 0 \le \chi^{(r)}(\tau) \le A_{max}^{(r)}, \quad \forall r \in \mathcal{R} \\ & (4.9), (4.10), (4.11).\end{aligned} \tag{4.23}$$

4.3 Online Distributed VNF Scheduling Algorithm

At time slot τ, we have two groups of decision variables which are separable in both the objective function and constraints of the instantaneous problem $\mathbf{P}_\tau$. One group is $\boldsymbol{\chi}(\tau)$, and the other group is $\boldsymbol{z}(\tau)$ and $\boldsymbol{S}(\tau)$. Thus, problem $\mathbf{P}_\tau$ is equivalent to two sub-problems, given by

$$\begin{aligned}\mathbf{P}_{\tau,1} : \max_{\boldsymbol{\chi}(\tau)} \quad & \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\Phi_1^{(r)}(\tau)|\boldsymbol{\Theta}(\tau)\right\} \\ \text{s.t.} \quad & 0 \le \chi^{(r)}(\tau) \le A_{max}^{(r)}, \quad \forall r \in \mathcal{R}\end{aligned} \tag{4.24}$$

and

$$\begin{aligned}\mathbf{P}_{\tau,2} : \max_{\boldsymbol{z}(\tau),\boldsymbol{S}(\tau)} \quad & \sum_{r\in\mathcal{R}} \mathbb{E}\left\{\Phi_2^{(r)}(\tau)|\boldsymbol{\Theta}(\tau)\right\} \\ \text{s.t.} \quad & (4.9), (4.10), (4.11).\end{aligned} \tag{4.25}$$

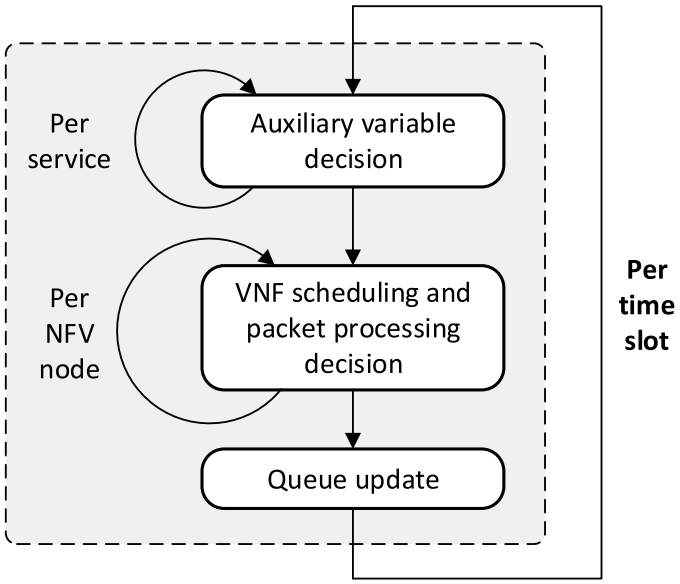


Figure 4.3: Flowchart of the proposed online distributed VNF scheduling algorithm.

Using the concept of opportunistically maximizing an expectation, the objective function of sub-problems $\mathbf{P}_{\tau,1}$ and $\mathbf{P}_{\tau,2}$ can be maximized by maximizing $\sum_{r\in\mathcal{R}} \Phi_1^{(r)}(\tau)$ or $\sum_{r\in\mathcal{R}} \Phi_2^{(r)}(\tau)$ given the observed values of $\mathbf{\Theta}(\tau)$ under the corresponding constraints [85]. We design an online distributed VNF scheduling algorithm to solve sub-problems $\mathbf{P}_{\tau,1}$ and $\mathbf{P}_{\tau,2}$ for each time slot, with a flowchart given in Fig. 4.3.

Auxiliary variable decision

Both the objective function and constraint of sub-problem $\mathbf{P}_{\tau,1}$ are separable among services. Hence, the problem can be further decomposed into service-level sub-problems which determine the auxiliary variable individually for each service. For service $r \in \mathcal{R}$, since only virtual queue length $F^{(r)}(\tau)$ and urgent packet queue lengths $\{Q_{h,\mathbb{U}}^{(r)}(\tau), \forall h \in \mathcal{H}_r\}$ among $\mathbf{\Theta}(\tau)$ appear in $\Phi_1^{(r)}(\tau)$, the optimal value of $\chi^{(r)}(\tau)$ can be derived by observing $F^{(r)}(\tau)$ and $\{Q_{h,\mathbb{U}}^{(r)}(\tau), \forall h \in \mathcal{H}_r\}$, and solving an optimization problem given by

$$\begin{aligned} \mathbf{P}_{\tau,1}^{(r)}: \max_{\chi^{(r)}(\tau)} \quad & \vartheta \cdot \phi(\chi^{(r)}(\tau)) - (P_r)^2 \left[F^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} Q_{h,\mathbb{U}}^{(r)}(\tau) \right] \chi^{(r)}(\tau) \\ \text{s.t.} \quad & 0 \le \chi^{(r)}(\tau) \le A_{max}^{(r)}. \end{aligned} \tag{4.26}$$

The optimal solution of problem $\mathbf{P}_{\tau,1}^{(r)}$, denoted by $\chi^{(r)*}(\tau)$, is derived by differentiating the objective function with respect to $\chi^{(r)}(\tau)$, and is given by

$$\chi^{(r)*}(\tau) = \begin{cases} A_{max}^{(r)}, & \text{if } (P_r)^2 \left[F^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} Q_{h,\mathbb{U}}^{(r)}(\tau)\right] \leq \frac{\vartheta}{A_{max}^{(r)}} \\ \frac{\vartheta}{(P_r)^2\left[F^{(r)}(\tau)+\sum_{h\in\mathcal{H}_r} Q_{h,\mathbb{U}}^{(r)}(\tau)\right]}, & \text{otherwise.} \end{cases} \tag{4.27}$$

VNF scheduling and packet processing

Let $S_{h,\mathbb{N}}^{(r)}(\tau)$ denote the number of non-urgent packets processed at VNF $V_h^{(r)}$ of service $r \in \mathcal{R}$ during time slot τ, with $S_h^{(r)}(\tau) = S_{h,\mathbb{N}}^{(r)}(\tau) + S_{h,\mathbb{U}}^{(r)}(\tau)$. Then, we can rewrite $\Phi_2^{(r)}(\tau)$ in (4.22) as

$$\Phi_2^{(r)}(\tau) = \sum_{h\in\mathcal{H}_r} \left[\omega_{h,\mathbb{U}}^{(r)}(\tau) S_{h,\mathbb{U}}^{(r)}(\tau) + \omega_{h,\mathbb{N}}^{(r)}(\tau) S_{h,\mathbb{N}}^{(r)}(\tau)\right], \quad \forall r \in \mathcal{R} \tag{4.28}$$

where $\omega_{h,\mathbb{U}}^{(r)}(\tau)$ and $\omega_{h,\mathbb{N}}^{(r)}(\tau)$ are the adaptive scheduling weights for one urgent packet and for one non-urgent packet at VNF $V_h^{(r)}$ during time slot τ respectively, given by

$$\omega_{h,\mathbb{U}}^{(r)}(\tau) = \left(\frac{\varphi}{\bar{a}_r}\right)^2 W^{(r)}(\tau) + (P_r)^2 F^{(r)}(\tau) - \left(P_{h+1}^{(r)}\right)^2 q_{h+1}^{(r)}(\tau) \tag{4.29}$$

$$\omega_{h,\mathbb{N}}^{(r)}(\tau) = \left(P_h^{(r)}\right)^2 q_h^{(r)}(\tau) - \left(P_{h+1}^{(r)}\right)^2 q_{h+1}^{(r)}(\tau). \tag{4.30}$$

Here, we overuse $P_{h+1}^{(r)}$ and $q_{h+1}^{(r)}$, with $P_{H_r+1}^{(r)} \equiv q_{H_r+1}^{(r)} \equiv 0$ for $h = H_r$. The scheduling weight for one urgent packet at VNF $V_h^{(r)}$, i.e., $\omega_{h,\mathbb{U}}^{(r)}(\tau)$, corresponds to the difference between the weighted virtual queue lengths of service $r \in \mathcal{R}$ and the weighted physical packet processing queue length at downstream VNF $V_{h+1}^{(r)}$, while the scheduling weight for one non-urgent packet at VNF $V_h^{(r)}$, i.e., $\omega_{h,\mathbb{N}}^{(r)}(\tau)$, corresponds to the weighted differential backlogs between VNF $V_h^{(r)}$ and downstream VNF $V_{h+1}^{(r)}$. Through such a differentiation between urgent and non-urgent packets, packet urgency and throughput performance are incorporated in VNF scheduling beyond the classical backpressure scheduling policy. A temporary greater congestion level in the virtual queues indicates less satisfaction or even violation of the service throughput requirement, resulting in a larger scheduling weight for each urgent packet in (4.29). A greater congestion level at the downstream VNF discourages

packet processing at the upstream VNF to avoid further worsening the congestion situation, through reducing the packet scheduling weights for both urgent and non-urgent packets at the upstream VNF. For the classical backpressure algorithm adapted for processing resources, no differentiation is considered between urgent and non-urgent packets, and all packets are treated as non-urgent packets with the same scheduling weight in (4.30).

The VNF scheduling and packet processing decisions of different services are coupled through the processing resource budget constraints at the NFV nodes. We rewrite $\sum_{r\in\mathcal{R}} \Phi_2^{(r)}(\tau)$ as a summation over the NFV nodes, given by

$$\sum_{r\in\mathcal{R}} \Phi_2^{(r)}(\tau) = \sum_{n\in\mathcal{N}} \sum_{(r,h)\in\mathcal{V}_n} \left[\omega_{h,\mathbb{U}}^{(r)}(\tau) S_{h,\mathbb{U}}^{(r)}(\tau) + \omega_{h,\mathbb{N}}^{(r)}(\tau) S_{h,\mathbb{N}}^{(r)}(\tau)\right]. \tag{4.31}$$

Since both the objective function and constraints of sub-problem $\mathbf{P}_{\tau,2}$ are separable among the NFV nodes, the problem is further decomposed into NFV node level sub-problems, given by

$$\begin{aligned} \mathbf{P}_{\tau,2}^{(n)}: \quad & \max_{\{z_h^{(r)}(\tau), S_{h,m}^{(r)}(\tau), \forall (r,h)\in\mathcal{V}_n, \forall m\in\mathcal{M}_h^{(r)}\}} && \sum_{(r,h)\in\mathcal{V}_n} \left[\omega_{h,\mathbb{U}}^{(r)}(\tau) S_{h,\mathbb{U}}^{(r)}(\tau) + \omega_{h,\mathbb{N}}^{(r)}(\tau) S_{h,\mathbb{N}}^{(r)}(\tau)\right] \\ & \text{s.t.} && (4.9), (4.10), (4.11) \text{ for } n\in\mathcal{N} \end{aligned} \tag{4.32}$$

for NFV node $n\in\mathcal{N}$. Let ${z_h^{(r)}}^*(\tau)$ be the optimal binary scheduling decision variable for VNF $V_h^{(r)}$ during time slot τ. The number of packets processed at VNF $V_h^{(r)}$ during time slot τ is $S_h^{(r)}(\tau) = {z_h^{(r)}}^*(\tau) \cdot \min\left(Q_h^{(r)}(\tau), \left\lfloor \frac{\sum_{n\in\mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)}} \right\rfloor\right)$ which satisfies constraints (4.9)-(4.11) and maximizes the objective function in (4.32). For VNF $V_h^{(r)}$, the urgent packets with residual lifetime $m = H_r - h + 1$ have the highest priority to be processed, followed by the non-urgent packets whose priority decreases in ascending order of residual lifetime, corresponding to a first-come-first-serve (FCFS) prioritization principle. Let $\hat{S}_{h,m}^{(r)}(\tau)$ be the number of packets with residual lifetime $m\in\mathcal{M}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ if VNF $V_h^{(r)}$ is scheduled, given by

$$\hat{S}_{h,m}^{(r)}(\tau) = \begin{cases} Q_{h,m}^{(r)}(\tau), & \text{if } m < m_0 \\ \min\left(Q_h^{(r)}(\tau), \left\lfloor \frac{\sum_{n\in\mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)}} \right\rfloor\right) - \sum_{m<m_0} Q_{h,m}^{(r)}(\tau), & \text{if } m = m_0 \\ 0, & \text{otherwise} \end{cases} \tag{4.33}$$

where $m_0\in\mathcal{M}_h^{(r)}$ satisfies

$$\sum_{m\le m_0} Q_{h,m}^{(r)}(\tau) \ge \min\left(Q_h^{(r)}(\tau), \left\lfloor \frac{\sum_{n\in\mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)}} \right\rfloor\right) > \sum_{m<m_0} Q_{h,m}^{(r)}(\tau). \tag{4.34}$$

Define a VNF scheduling weight $\omega_h^{(r)}(\tau)$ for VNF $V_h^{(r)}$ during time slot τ, given by

$$\omega_h^{(r)}(\tau) = \omega_{h,\mathbb{U}}^{(r)}(\tau)\hat{S}_{h,H_r-h+1}^{(r)}(\tau) + \omega_{h,\mathbb{N}}^{(r)}(\tau) \sum_{m>H_r-h+1} \hat{S}_{h,m}^{(r)}(\tau), \quad \forall r \in \mathcal{R},\ h \in \mathcal{H}_r \tag{4.35}$$

which is a summation of the total packet scheduling weights for all the urgent and non-urgent packets that are processed at VNF $V_h^{(r)}$ if VNF $V_h^{(r)}$ is scheduled during time slot τ, where the per-packet scheduling weights for each urgent packet and each non-urgent packet are given in (4.29) and (4.30) respectively. For the classical backpressure algorithm adapted for processing resources without packet urgency awareness, the VNF scheduling weight for VNF $V_h^{(r)}$ at time slot τ is simplified as $\omega_{h,\mathbb{N}}^{(r)}(\tau) \min\left(Q_h^{(r)}(\tau), \left\lfloor \frac{\sum_{n\in\mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)}} \right\rfloor\right)$. Then, the VNF with the largest VNF scheduling weight is greedily scheduled at each NFV node, and the optimal solutions for problem $\mathbf{P}_{\tau,2}^{(n)}$ associated with NFV node $n \in \mathcal{N}$ are given by

$$z_h^{(r)^*}(\tau) = \mathbb{1}\{(r,h) = \arg\max_{(r,h)\in\mathcal{V}_n} \omega_h^{(r)}(\tau)\}, \quad \forall (r,h) \in \mathcal{V}_n \tag{4.36}$$

$$S_{h,m}^{(r)^*}(\tau) = z_h^{(r)^*}(\tau)\hat{S}_{h,m}^{(r)}(\tau), \quad \forall (r,h) \in \mathcal{V}_n,\ \forall m \in \mathcal{M}_h^{(r)}. \tag{4.37}$$

For a service, a temporal throughput degradation below the minimum requirement results in a higher congestion level in the virtual queues and more urgent packets in the packet processing queues, which in turn increases the VNF scheduling weights for all VNFs in the service. In this way, the VNFs have more chances to be scheduled, leading to an improvement in the throughput.

Queue updates

Combining all the decisions for time slot τ, the queue backlogs for time slot $\tau+1$ including the physical packet processing queue lengths, $\{q_h^{(r)}(\tau+1), \forall r, h\}$, the virtual packet processing queue lengths, $\{Q_{h,m}^{(r)}(\tau+1), \forall r, h, m\}$, and the service-level virtual queue lengths, $\{W^{(r)}(\tau+1), F^{(r)}(\tau+1), \forall r\}$, are updated according to (4.1), (4.4), (4.15) and (4.16).

Performance optimality

The proposed online VNF scheduling algorithm achieves $\mathcal{O}(\frac{1}{\vartheta})$ near-optimal total utility, with the optimality gap decreasing with ϑ, and results in linearly increasing total

queue backlogs with the increase of ϑ, demonstrating an $\left[\mathcal{O}(\frac{1}{\vartheta}), \mathcal{O}(\vartheta)\right]$ utility-backlog trade-off [85].

4.4 VNF Scheduling Algorithm with Packet Rushing

Consider an extreme case under the assumption of no packet rushing. For a service with H_r VNFs in the chain, a packet experiences at least an E2E delay of $H_r T$ even if there is no packet queueing at all the VNFs, where T is the time slot length. Such a delay overhead is referred to as the worst-case E2E delay overhead, which is non-negligible for a realistic time slot length such as $1ms$ and a short E2E delay such as $10ms$. However, if packet rushing is allowed, when a scheduled VNF $V_h^{(r)}$ $(h > 1)$ is unsaturated for time slot τ, i.e., there are residual CPU cycles before the end of the time slot after all the packets in its queue are processed, corresponding to the condition of $S_h^{(r)}(\tau) = Q_h^{(r)}(\tau) < \left\lfloor \frac{\sum_{n \in \mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)}} \right\rfloor$, some packets processed by upstream VNF $V_{h-1}^{(r)}$ during time slot τ can be further processed by VNF $V_h^{(r)}$ using the residual CPU cycles during the same time slot, hence enhancing resource utilization and reducing packet E2E delay. Such extra packets processed by VNF $V_h^{(r)}$ are referred to as rushing packets for VNF $V_h^{(r)}$. The packets which are originally in the queue are referred to as non-rushing packets. The number of non-rushing packets processed at VNF $V_h^{(r)}$ during time slot τ is $S_h^{(r)}(\tau)$. Since it is possible that a packet can be processed by several consecutive VNFs during one time slot, the rushing packets processed by VNF $V_h^{(r)}$ can include both rushing and non-rushing packets processed by upstream VNF $V_{h-1}^{(r)}$.

4.4.1 Packet Rushing Analysis

Assume that the VNF scheduling variables, i.e., $\{z_h^{(r)}(\tau), \forall h \in \mathcal{H}_r\}$, and the number of non-rushing packets processed at each VNF, i.e., $\{S_h^{(r)}(\tau), \forall h \in \mathcal{H}_r\}$, are given for service $r \in \mathcal{R}$ during time slot τ. We analyze the number of rushing packets processed by each VNF of service $r \in \mathcal{R}$ during time slot τ, denoted by $R_h^{(r)}(\tau)$ for VNF $V_h^{(r)}$. The VNFs of service $r \in \mathcal{R}$ are classified into three subsets for time slot τ, given by

$$\mathcal{H}_1^{(r)}(\tau) = \{h \in \mathcal{H}_r | z_h^{(r)}(\tau) = 0\}, \quad r \in \mathcal{R} \tag{4.38}$$

$$\mathcal{H}_2^{(r)}(\tau) = \{h \in \mathcal{H}_r \backslash \{1\} | z_h^{(r)}(\tau) = 1, z_{h-1}^{(r)}(\tau) = 1, S_h^{(r)}(\tau) < \left\lfloor \frac{\sum_{n \in \mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)}} \right\rfloor \}, \ r \in \mathcal{R} \tag{4.39}$$

$$\mathcal{H}_3^{(r)}(\tau) = \mathcal{H}_r \backslash \left(\mathcal{H}_1^{(r)}(\tau) \cup \mathcal{H}_2^{(r)}(\tau) \right), \quad r \in \mathcal{R}. \tag{4.40}$$

For service $r \in \mathcal{R}$, subset $\mathcal{H}_1^{(r)}(\tau)$ includes all the unscheduled VNFs during time slot τ, subset $\mathcal{H}_2^{(r)}(\tau)$ includes all the scheduled VNFs where there is packet rushing opportunity, i.e., the scheduled unsaturated VNFs whose upstream VNF is also scheduled, and subset $\mathcal{H}_3^{(r)}(\tau)$ includes all the scheduled VNFs where there is no packet rushing. Intuitively, the overall packet rushing opportunity is higher, if we have more VNFs in subset $\mathcal{H}_2^{(r)}(\tau)$ with more residual resources. Let $\tilde{S}_h^{(r)}(\tau)$ be the actual number of packets processed by VNF $V_h^{(r)}$ during time slot τ with the consideration of packet rushing, given by

$$\tilde{S}_h^{(r)}(\tau) = S_h^{(r)}(\tau) + R_h^{(r)}(\tau), \quad \forall r \in \mathcal{R}, \ h \in \mathcal{H}_r. \tag{4.41}$$

We have $\tilde{S}_h^{(r)}(\tau) = S_h^{(r)}(\tau), \forall h \in \mathcal{H}_1^{(r)}(\tau) \cup \mathcal{H}_3^{(r)}(\tau)$ and $\tilde{S}_h^{(r)}(\tau) \geq S_h^{(r)}(\tau), \forall h \in \mathcal{H}_2^{(r)}(\tau)$ for service $r \in \mathcal{R}$ during time slot τ.

We consider a finite timeline, $\mathtt{t} \in [0, T]$ starting at the beginning of time slot τ and ending at the end of time slot τ, where T is the time slot length in second. Assume that the scheduled VNFs during time slot τ start to process the first packet in their queues at time instant $t = 0$. Let $\upsilon_{h,max}^{(r)}$ denote the maximum packet processing rate (in packet/s) for VNF $V_h^{(r)}$, given by

$$\upsilon_{h,max}^{(r)} = \frac{\sum_{n \in \mathcal{N}} \mathrm{x}_n^{rh} C_n}{P_h^{(r)} T}, \quad \forall r \in \mathcal{R}, \ \forall h \in \mathcal{H}_r. \tag{4.42}$$

For VNF $V_h^{(r)}$ in subset $\mathcal{H}_2^{(r)}(\tau)$, there are $S_h^{(r)}(\tau)$ non-rushing packets being processed first in the maximum packet processing rate $\upsilon_{h,max}^{(r)}$. Then, the rushing packets from the upstream VNF $V_{h-1}^{(r)}$ start to be processed at VNF $V_h^{(r)}$. However, the actual processing rate for the rushing packets depend on the packet processing rate of either VNF $V_h^{(r)}$ or VNF $V_{h-1}^{(r)}$ in different conditions. Let $\upsilon_h^{(r)}(\mathtt{t})$ denote the actual packet processing rate (in packet/s) for VNF $V_h^{(r)}$ at time $\mathtt{t} \in [0, T]$, with $\upsilon_h^{(r)}(0) = \upsilon_{h,max}^{(r)}$. There are three cases for $\upsilon_h^{(r)}(\mathtt{t})$, depending on the relationships among $\upsilon_{h,max}^{(r)}$, $\upsilon_{h-1}^{(r)}(\mathtt{t})$, $S_h^{(r)}(\tau)$ and T. Fig. 4.4 and Fig. 4.5 illustrate the three cases for $\upsilon_h^{(r)}(\mathtt{t})$, with $\upsilon_{h-1}^{(r)}(\mathtt{t})$ being either a constant or a decreasing step function with t, respectively.

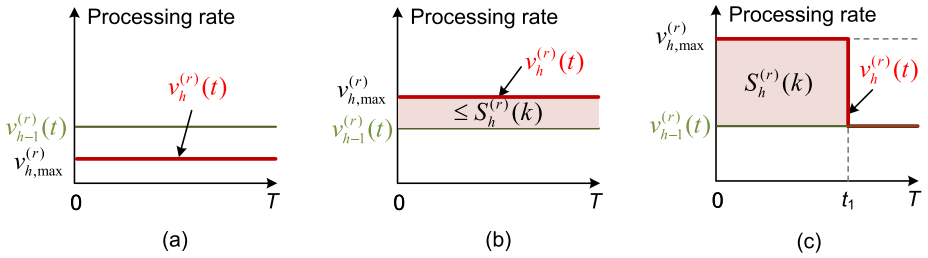


Figure 4.4: An illustration of actual packet processing rate $v_h^{(r)}(\mathsf{t})$ if $v_{h-1}^{(r)}(\mathsf{t})$ is a constant.

- Case 1: If VNF $V_h^{(r)}$ cannot process packets faster than VNF $V_{h-1}^{(r)}$ within time duration T, i.e., $v_{h,max}^{(r)} \leq \min_{\mathsf{t}\in[0,T]} v_{h-1}^{(r)}(\mathsf{t})$, we have $v_h^{(r)}(\mathsf{t}) = v_{h,max}^{(r)}, \forall \mathsf{t} \in [0,T]$, as illustrated in Fig. 4.4(a) and Fig. 4.5(a).

- Case 2: Under the condition that $v_{h,max}^{(r)} > \min_{\mathsf{t}\in[0,T]} v_{h-1}^{(r)}(\mathsf{t})$, if the number of packets processed by VNF $V_h^{(r)}$ in the maximum processing rate $v_{h,max}^{(r)}$ within time duration T does not exceed the number of packets processed by VNF $V_{h-1}^{(r)}$ within time duration T by more than $S_h^{(r)}(\tau)$, i.e., $v_{h,max}^{(r)}T - \int_0^T v_{h-1}^{(r)}(\mathsf{t})d\mathsf{t} \leq S_h^{(r)}(\tau)$, the actual processing rate for both the non-rushing and rushing packets at VNF $V_h^{(r)}$ is equal to $v_{h,max}^{(r)}$ within time duration T, as illustrated in Fig. 4.4(b) and Fig. 4.5(b).

- Case 3: Under the condition that $v_{h,max}^{(r)} > \min_{\mathsf{t}\in[0,T]} v_{h-1}^{(r)}(\mathsf{t})$, if VNF $V_h^{(r)}$ can process more packets than VNF $V_{h-1}^{(r)}$ by $S_h^{(r)}(\tau)$ at a certain time instant $\mathsf{t}_1 < T$, i.e., $v_{h,max}^{(r)}\mathsf{t}_1 - \int_0^{\mathsf{t}_1} v_{h-1}^{(r)}(\mathsf{t})d\mathsf{t} = S_h^{(r)}(\tau)$, the actual packet processing rate for VNF $V_h^{(r)}$ is equal to that of the upstream VNF $V_{h-1}^{(r)}$ after time t_1, as illustrated in Fig. 4.4(c) and Fig. 4.5(c). We refer to time instant t_1 as transition time instant.

We see that the actual packet processing rate $v_h^{(r)}(\mathsf{t})$ is either a constant or a decreasing step function with t, no matter $v_{h-1}^{(r)}(\mathsf{t})$ is a constant or a decreasing step function with t. The number of rushing packets processed by VNF $V_h^{(r)}$ in subset $\mathcal{H}_2^{(r)}(\tau)$ during time slot τ, i.e., $R_h^{(r)}(\tau)$, is limited by the actual number of packets processed by the upstream VNF $V_{h-1}^{(r)}$ during time slot τ and the maximum number of extra packets that VNF $V_h^{(r)}$ can process in the actual packet processing rate $v_h^{(r)}(\mathsf{t})$ within time duration T, represented by

$$R_h^{(r)}(\tau) = \min\left(\tilde{S}_{h-1}^{(r)}(\tau), \int_0^T v_h^{(r)}(\mathsf{t})d\mathsf{t} - S_h^{(r)}(\tau)\right). \tag{4.43}$$

Algorithm 5: Packet rushing analysis for service $r \in \mathcal{R}$ during time slot τ

Input: $\{S_h^{(r)}(\tau), h \in \mathcal{H}_r\}$, sets $\mathcal{H}_1^{(r)}(\tau)$, $\mathcal{H}_2^{(r)}(\tau)$, $\mathcal{H}_3^{(r)}(\tau)$.
Initialize: $\{R_h^{(r)}(\tau) = 0, h \in \mathcal{H}_r\}$.
for $h = 1, \cdots, H_r$ **do**
  if $h \in \mathcal{H}_3^{(r)}(\tau)$, **then**
    $\boldsymbol{\upsilon}_h^{(r)} = \upsilon_{h,max}^{(r)}$, $\mathbf{t}_h^{(r)} = [0, T]^\intercal$.
  end
  if $h \in \mathcal{H}_2^{(r)}(\tau)$ **then**
    if $\upsilon_{h,max}^{(r)} \leq \min\left(\boldsymbol{\upsilon}_{h-1}^{(r)}\right)$ **then**
      $\boldsymbol{\upsilon}_h^{(r)} = \upsilon_{h,max}^{(r)}$, $\mathbf{t}_h^{(r)} = [0, T]^\intercal$.
    else
      if $\upsilon_{h,max}^{(r)} T - {\boldsymbol{\upsilon}_{h-1}^{(r)}}^\intercal \cdot \left(\mathbf{t}_{h-1}^{(r)}\left[2 : |\mathbf{t}_{h-1}^{(r)}|\right] - \mathbf{t}_{h-1}^{(r)}\left[1 : |\boldsymbol{\upsilon}_{h-1}^{(r)}|\right]\right) \leq S_h^{(r)}(\tau)$ **then**
        $\boldsymbol{\upsilon}_h^{(r)} = \upsilon_{h,max}^{(r)}$, $\mathbf{t}_h^{(r)} = [0, T]^\intercal$.
      else
        $\boldsymbol{\delta} = \left(\upsilon_{h,max}^{(r)} \mathsf{e}_{|\boldsymbol{\upsilon}_{h-1}^{(r)}|} - \boldsymbol{\upsilon}_{h-1}^{(r)}\right) \circ \left(\mathbf{t}_{h-1}^{(r)}\left[2 : |\mathbf{t}_{h-1}^{(r)}|\right] - \mathbf{t}_{h-1}^{(r)}\left[1 : |\boldsymbol{\upsilon}_{h-1}^{(r)}|\right]\right)$.
        Find $\mathtt{j}_0$ with $\sum_{\mathtt{j} \leq \mathtt{j}_0} \boldsymbol{\delta}(\mathtt{j}) > S_h^{(r)}(\tau) \geq \sum_{\mathtt{j} < \mathtt{j}_0} \boldsymbol{\delta}(\mathtt{j})$.
        Calculate transition time instant $\mathtt{t}_1 = \mathbf{t}_{h-1}^{(r)}(\mathtt{j}_0 + 1) - \frac{\sum_{\mathtt{j} \leq \mathtt{j}_0} \boldsymbol{\delta}(\mathtt{j}) - S_h^{(r)}(\tau)}{\upsilon_{h,max}^{(r)} - \boldsymbol{\upsilon}_{h-1}^{(r)}(\mathtt{j}_0)}$.
        $\boldsymbol{\upsilon}_h^{(r)} = \left[\upsilon_{h,max}^{(r)}, \boldsymbol{\upsilon}_{h-1}^{(r)}\left[\mathtt{j}_0 : |\boldsymbol{\upsilon}_{h-1}^{(r)}|\right]^\intercal\right]^\intercal$.
        $\mathbf{t}_h^{(r)} = \left[0, \mathtt{t}_1, \mathbf{t}_{h-1}^{(r)}\left[\mathtt{j}_0 + 1 : |\mathbf{t}_{h-1}^{(r)}|\right]^\intercal\right]^\intercal$.
      end
    end
    $R_h^{(r)}(\tau) = \min\left(\tilde{S}_{h-1}^{(r)}(\tau), \left\lfloor {\boldsymbol{\upsilon}_h^{(r)}}^\intercal \cdot \left(\mathbf{t}_h^{(r)}\left[2 : |\mathbf{t}_h^{(r)}|\right] - \mathbf{t}_h^{(r)}\left[1 : |\boldsymbol{\upsilon}_h^{(r)}|\right]\right)\right\rfloor - S_h^{(r)}(\tau)\right)$.
  end
  Calculate $\tilde{S}_h^{(r)}(\tau)$ according to (4.41).
end
Output: $\{R_h^{(r)}(\tau), h \in \mathcal{H}_r\}$, $\{\tilde{S}_h^{(r)}(\tau), h \in \mathcal{H}_r\}$.

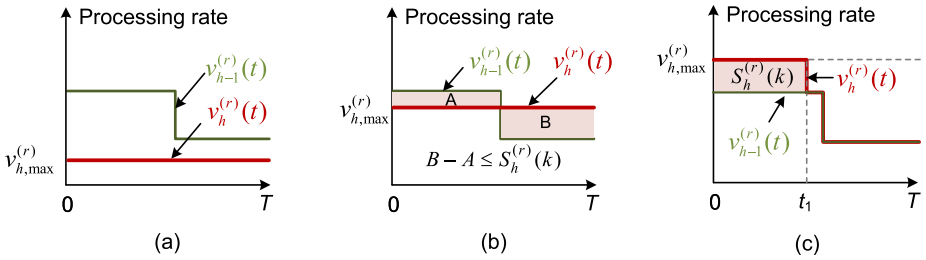


Figure 4.5: An illustration of actual packet processing rate $v_h^{(r)}(\mathtt{t})$ if $v_{h-1}^{(r)}(\mathtt{t})$ is a decreasing step function.

For service $r \in \mathcal{R}$, packet rushing analysis is performed iteratively for VNFs from source to destination, with a procedure given in Algorithm 5. In the algorithm, we use two vectors, $\boldsymbol{v}_h^{(r)}$ and $\mathbf{t}_h^{(r)}$, to represent function $v_h^{(r)}(\mathtt{t}), \mathtt{t} \in [0, T]$. Let $\boldsymbol{v}_h^{(r)}$ be a vector of actual packet processing rates for VNF $V_h^{(r)}$, e.g., $\boldsymbol{v}_h^{(r)} = [100, 80, 60]^\intercal$, and let $\mathbf{t}_h^{(r)}$ be a vector of time boundaries between the actual packet processing rates, e.g., $\mathbf{t}_h^{(r)} = [0, \frac{T}{2}, \frac{3T}{4}, T]^\intercal$, where superscript $^\intercal$ denotes the transpose operator. The dimension of vector $\mathbf{t}_h^{(r)}$ is larger than that of vector $\boldsymbol{v}_h^{(r)}$ by 1, i.e., $|\mathbf{t}_h^{(r)}| = |\boldsymbol{v}_h^{(r)}| + 1$. The relationship between $v_h^{(r)}(\mathtt{t})$ and the two vectors is given by

$$v_h^{(r)}(\mathtt{t}) = \boldsymbol{v}_h^{(r)}(\mathtt{j}), \quad \text{if } \mathbf{t}_h^{(r)}(\mathtt{j}) \le t < \mathbf{t}_h^{(r)}(\mathtt{j}+1) \tag{4.44}$$

where $\mathtt{j}$ is the index of the $\mathtt{j}$-th ($\mathtt{j} \le |\boldsymbol{v}_h^{(r)}|$) element in vector $\boldsymbol{v}_h^{(r)}$ or $\mathbf{t}_h^{(r)}$. Then, we have

$$\int_0^T v_h^{(r)}(\mathtt{t}) d\mathtt{t} = \boldsymbol{v}_h^{(r)\intercal} \cdot \left(\mathbf{t}_h^{(r)} \left[2 : |\mathbf{t}_h^{(r)}| \right] - \mathbf{t}_h^{(r)} \left[1 : |\boldsymbol{v}_h^{(r)}| \right] \right). \tag{4.45}$$

In Algorithm 5, lines 7-8 correspond to Case 1 for $v_h^{(r)}(\mathtt{t})$ of VNF $V_h^{(r)}$ in subset $\mathcal{H}_2^{(r)}(\tau)$, lines 10-11 correspond to Case 2, and lines 12-17 correspond to Case 3. In line 13, $\mathbf{e}_{|\boldsymbol{v}_{h-1}^{(r)}|}$ is a vector of length $|\boldsymbol{v}_{h-1}^{(r)}|$ with every element equal to 1, and $\circ$ represents the element-wise product operation between two vectors. Index $\mathtt{j}_0$ in line 14 is the smallest index satisfying $v_{h,max}^{(r)} \mathbf{t}_{h-1}^{(r)}(\mathtt{j}_0 + 1) - \int_0^{\mathbf{t}_{h-1}^{(r)}(\mathtt{j}_0+1)} v_{h-1}^{(r)}(\mathtt{t}) d\mathtt{t} > S_h^{(r)}(\tau)$.

4.4.2 Modified VNF Scheduling Algorithm

By taking advantage of packet rushing, a packet with residual lifetime $1 \le m < H_r - h + 1$ at VNF $V_h^{(r)}$ has opportunity to be successfully delivered before expiry, and a packet with

$m = M_r$ at VNF $V_1^{(r)}$ has opportunity to rush through all the VNFs in a service to the egress edge switch in one time slot. Hence, the set of packet residual lifetime at VNF $V_h^{(r)}$, denoted by $\tilde{\mathcal{M}}_h^{(r)}$, is modified to

$$\tilde{\mathcal{M}}_h^{(r)} = \{1, \cdots, M_r\}, \quad \forall r \in \mathcal{R},\ h \in \mathcal{H}_r. \tag{4.46}$$

However, in the worst case, a packet with residual lifetime $1 \leq m < H_r - h + 1$ at VNF $V_h^{(r)}$ cannot be timely delivered if there is no packet rushing opportunity. As discussed in Subsection 4.4.1, whether packet rushing can happen at a certain VNF or not during a given time slot is unknown until the VNF scheduling and packet processing decisions for the time slot are given. Hence, the VNF scheduling algorithm makes worst-case decisions under the assumption of no packet rushing in each time slot, and determines the number of non-rushing packets processed at each VNF. In the worst case, all the packets with residual lifetime of $1 \leq m \leq H_r - h + 1$ at VNF $V_h^{(r)}$ are urgent packets since they will be eventually dropped without any packet rushing opportunity if not processed in the current time slot, and other packets are non-urgent packets. Accordingly, we have $Q_{h,\mathbb{U}}^{(r)}(\tau) = \sum_{m=1}^{H_r-h+1} Q_{h,m}^{(r)}(\tau)$ and $S_{h,\mathbb{U}}^{(r)}(\tau) = \sum_{m=1}^{H_r-h+1} S_{h,m}^{(r)}(\tau)$.

The modified VNF scheduling algorithm is derived based on the worst-case Lyapunov drift-plus-penalty, in which the physical and virtual queue lengths are updated according to (4.1), (4.15) and (4.16) with the new definitions of $Q_{h,\mathbb{U}}^{(r)}(\tau)$ and $S_{h,\mathbb{U}}^{(r)}(\tau)$. We use $D_h^{(r)}(\tau) = Q_{h,\mathbb{U}}^{(r)}(\tau) - S_{h,\mathbb{U}}^{(r)}(\tau)$ as the worst-case number of dropped packets at VNF $V_h^{(r)}$ in (4.1). In the modified algorithm, the auxiliary variable decision is the same as that without packet rushing, except for using the new definition of $Q_{h,\mathbb{U}}^{(r)}(\tau)$. For VNF scheduling and packet processing, we consider an FCFS prioritization principle for the packets with different residual lifetime at the scheduled VNFs. If VNF $V_h^{(r)}$ is scheduled, the number of non-rushing packets with residual lifetime $m \in \tilde{\mathcal{M}}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ, i.e., $\hat{S}_{h,m}^{(r)}(\tau)$, is given by (4.33) where $m_0 \in \tilde{\mathcal{M}}_h^{(r)}$ satisfies (4.34). Then, the optimal VNF scheduling and packet processing decisions are made based on a modified VNF scheduling weight, $\tilde{\omega}_h^{(r)}(\tau)$, given by

$$\tilde{\omega}_h^{(r)}(\tau) = \omega_{h,\mathbb{U}}^{(r)}(\tau) \sum_{m=1}^{H_r-h+1} \hat{S}_{h,m}^{(r)}(\tau) + \omega_{h,\mathbb{N}}^{(r)}(\tau) \sum_{m>H_r-h+1} \hat{S}_{h,m}^{(r)}(\tau), \quad \forall r \in \mathcal{R},\ h \in \mathcal{H}_r. \tag{4.47}$$

Correct queue updates

After all decisions for time slot τ are made using the modified VNF scheduling algorithm, packet rushing analysis is performed for each service. Although the modified algorithm is derived based on the worst-case queue length updates, the true queue lengths can be updated at the end of time slot τ. The true physical queue length evolution equations are

$$\begin{aligned} q_h^{(r)}(\tau+1) = &\left[q_h^{(r)}(\tau) - S_h^{(r)}(\tau) - D_h^{(r)}(\tau)\right]^+ + \left[\tilde{S}_{h-1}^{(r)}(\tau) - R_h^{(r)}(\tau)\right] \mathbb{1}\{h>1\}, \\ &+ A^{(r)}(\tau)\mathbb{1}\{h=1\}, \quad \forall r \in \mathcal{R},\ \forall h \in \mathcal{H}_r \end{aligned} \tag{4.48}$$

where $D_h^{(r)}(\tau)$ is updated as $D_h^{(r)}(\tau) = \left[Q_{h,1}^{(r)}(\tau) - S_{h,1}^{(r)}(\tau)\right]^+$, since only the packets with residual lifetime $m = 1$ are actually dropped at VNF $V_h^{(r)}$ if they are not processed. Correspondingly, the true virtual queue length evolution equations in (4.15) and (4.16) are updated with $Q_{h,\mathbb{U}}^{(r)}(\tau) = Q_{h,1}^{(r)}(\tau)$ and $S_{h,\mathbb{U}}^{(r)}(\tau) = S_{h,1}^{(r)}(\tau)$. Let $R_{h,m}^{(r)}(\tau)$ and $\tilde{S}_{h,m}^{(r)}(\tau)$ be the number of rushing packets with residual lifetime $m \in \tilde{\mathcal{M}}_h^{(r)}$ and the actual total number of packets with residual lifetime $m \in \tilde{\mathcal{M}}_h^{(r)}$ that are processed at VNF $V_h^{(r)}$ during time slot τ respectively, with $\tilde{S}_{h,m}^{(r)}(\tau) = S_{h,m}^{(r)}(\tau) + R_{h,m}^{(r)}(\tau)$. For VNF $V_h^{(r)}$ with $R_h^{(r)}(\tau) = 0$, we have $R_{h,m}^{(r)}(\tau) = 0$ and $\tilde{S}_{h,m}^{(r)}(\tau) = S_{h,m}^{(r)}(\tau)$ for $\forall m \in \tilde{\mathcal{M}}_h^{(r)}$. For VNF $V_h^{(r)}$ with $R_h^{(r)}(\tau) = \sum_{m\in\tilde{\mathcal{M}}_h^{(r)}} R_{h,m}^{(r)}(\tau) > 0$, the rushing packets arrive at VNF $V_h^{(r)}$ in ascending order of packet residual lifetime. Thus, $R_{h,m}^{(r)}(\tau)$ is given by

$$R_{h,m}^{(r)}(\tau) = \begin{cases} \tilde{S}_{h-1,m}^{(r)}(\tau), & \text{if } 1 \le m < m_1 \\ R_h^{(r)}(\tau) - \sum_{m<m_1} \tilde{S}_{h-1,m}^{(r)}(\tau), & \text{if } m = m_1 \\ 0, & \text{otherwise} \end{cases} \tag{4.49}$$

where $m_1 \in \tilde{\mathcal{M}}_h^{(r)}$ satisfies $\sum_{m=1}^{m_1} \tilde{S}_{h-1,m}^{(r)}(\tau) \ge R_h^{(r)}(\tau) > \sum_{m<m_1} \tilde{S}_{h-1,m}^{(r)}(\tau)$. With packet rushing, the queueing dynamics of the delay-aware virtual packet processing queues are updated as

$$\begin{aligned} &Q_{1,M_r}^{(r)}(\tau+1) = A^{(r)}(\tau), && \forall r \in \mathcal{R} \\ &Q_{1,m}^{(r)}(\tau+1) = \left[Q_{1,m+1}^{(r)}(\tau) - S_{1,m+1}^{(r)}(\tau)\right]^+, && \forall r \in \mathcal{R}, \quad \forall m \in \tilde{\mathcal{M}}_1^{(r)} \backslash \{M_r\} \\ &Q_{h,M_r}^{(r)}(\tau+1) = 0, && \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r \backslash \{1\} \\ &Q_{h,M_r-1}^{(r)}(\tau+1) = \tilde{S}_{h-1,M_r}^{(r)}(\tau) - R_{h,M_r}^{(r)}(\tau), && \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r \backslash \{1\} \end{aligned}$$

Table 4.1: Simulation settings for virtual network topology

Topology	Services
1	Service 1: $n_1 \to n_2 \to n_3 \to n_7$; Service 2: $n_4 \to n_1 \to n_7 \to n_2$ Service 3: $n_8 \to n_5 \to n_2 \to n_6$; Service 4: $n_3 \to n_9 \to n_6 \to n_1$ Service 5: $n_5 \to n_6 \to n_4 \to n_3$; Service 6: $n_9 \to n_7 \to n_8 \to n_2$
2	All services: $n_1 \to n_2 \to n_3 \to n_4$

Table 4.2: Traffic sets for VNF scheduling simulation

Set	Service 1	Service 2	Service 3	Service 4	Service 5	Service 6
1	Trace 1 (1)	Trace 2 (1)	Trace 3 (1)	Trace 4 (1)	Trace 5 (1)	Trace 6 (1)
2	Trace 1 (1)	Trace 1 (4)	Trace 2 (1)	Trace 2 (4)	Trace 3 (1)	Trace 3 (4)

$$Q_{h,m}^{(r)}(\tau+1) = \left[Q_{h,m+1}^{(r)}(\tau) - S_{h,m+1}^{(r)}(\tau)\right]^+ + [\tilde{S}_{h-1,m+1}^{(r)}(\tau) - R_{h,m+1}^{(r)}(\tau)], \quad \forall r \in \mathcal{R}, \quad \forall h \in \mathcal{H}_r \backslash \{1\}, \quad \forall m \in \tilde{\mathcal{M}}_h^{(r)} \backslash \{M_r - 1, M_r\}. \tag{4.50}$$

4.5 Performance Evaluation

We consider two virtual network topologies, both with 6 services of given VNF placement at NFV nodes, as shown in Table 4.1, where n_i denotes the i-th NFV node. The services in topology 1 traverse through different virtual paths in a network of 9 NFV nodes, while all the services in topology 2 share a common virtual path through 4 NFV nodes. The packet E2E deadline of each service is set as $10ms$. We assume that the maximum packet dropping ratios for different services are the same, denoted by ε. By default, ε is set as 10^{-3}. We use 6 real-world stationary traffic traces with packet timestamp information [20, 67]. The average packet arrival rates of the 6 traffic traces are 17915, 25627, 33038, 51182, 47810, 67912 in packet/s respectively. We consider two traffic sets for the services, as given in Table 4.2, where the number inside the bracket indicates the processing density in kilo-cycle per packet (i.e., $Kcpp$). For example, in traffic set 1, we use traffic trace 1 for service 1, with a processing density of $1Kcpp$ for each VNF in the service [50]. In traffic set 1, the

Table 4.3: Default parameters in VNF scheduling

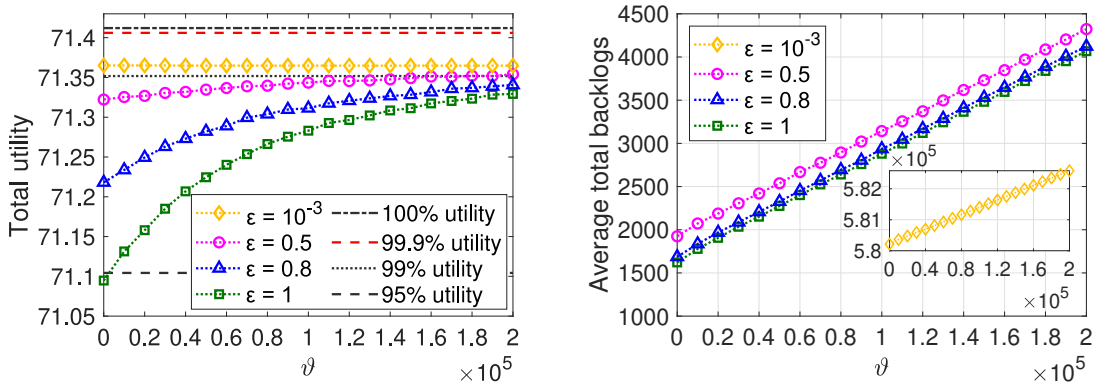
M_r	Packet E2E deadline	$10ms$
T	Time slot length	$1ms$
ε	Maximum packet dropping ratio	10^{-3}
$P_h^{(r)}$	Processing density	$1Kcpp$
ϑ	Utility importance parameter	10^5
Γ	Total number of time slots	10^5

services have different traffic traces and the same processing density. In traffic set 2, each traffic trace is used for two services with different processing densities. We use topology 1 and traffic set 1 as the default simulation setting. The time slot length T is set as $1ms$ by default. The total processing resource budget (in cycle per time slot) at NFV node $n \in \mathcal{N}$ is proportional to the average processing resource demand of all the VNFs placed at the NFV node, given by

$$C_n = \rho \sum_{(r,h)\in\mathcal{V}_n} \bar{a}_r P_h^{(r)}, \quad \forall n \in \mathcal{N} \tag{4.51}$$

where ρ is referred to as the resource overprovisioning ratio. The utility importance parameter, ϑ, is set as 10^5 by default. With a certain simulation setting, let the VNF scheduling algorithm run for $\Gamma = 10^5$ time slots. The performance metrics such as the throughput $\bar{f}_r$ in (4.6) and the average E2E delay $\bar{d}_r$ in (4.5) are calculated based on the average over the Γ time slots. All the default parameters for performance evaluation are summarized in Table 4.3.

We first evaluate the performance of the proposed basic VNF scheduling algorithm without packet rushing. The utility-backlog trade-off is investigated by increasing the utility importance parameter ϑ from 1 to 200000. The resource overprovisioning ratio is set as $\rho = 3$. We also examine the impact of QoS constraints on the utility-backlog trade-off, by setting $\varepsilon = 10^{-3}, 0.5, 0.8, 1$ to represent different levels of relaxation on the QoS constraints. For $\varepsilon = 1$, it corresponds to a utility maximization problem without explicit QoS constraints for each service. Fig. 4.6(a) shows the total utility with the increase of ϑ at different values of ε. In the figure, the "x% utility" represents the total utility when the timely delivery ratio of each service is x%. With the increase of ϑ, the total utility with $\varepsilon = 10^{-3}$ is stable, which is slightly beyond the 99% utility but does not reach the



(a) Total utility

(b) Average total backlogs

Figure 4.6: Trade-off between total utility and average total backlogs with respect to ϑ ($\rho = 3$).

99.9% utility, inferring that a resource overprovisioning ratio of $\rho = 3$ is not sufficient for such a strict QoS requirement without packet rushing under the given simulation settings. With a relaxed QoS constraint, i.e., $\varepsilon \in \{0.5, 0.8, 1\}$, the total utility gradually increases and gets closer to the 99% utility with the increase of ϑ. We notice that the achieved total utility without explicit QoS constraints (i.e., $\varepsilon = 1$) at $\vartheta = 1$ is close to the 95% utility. However, for each value of ε, we see an linear increase in the average total backlogs (i.e., average of the total actual and virtual queue lengths over time slots) with the increase of ϑ in Fig. 4.6(b), demonstrating an $\left[\mathcal{O}(\frac{1}{\vartheta}), \mathcal{O}(\vartheta)\right]$ utility-backlog trade-off with the increase of ϑ. We also observe that more utility is achieved with a more strict QoS constraint for a certain value of ϑ, at a cost of a greater congestion level. The average total backlogs with $\varepsilon = 10^{-3}$ is even two orders of magnitude higher than that with relaxed QoS constraints, since the QoS constraint violation with $\varepsilon = 10^{-3}$ results in unstable virtual queue $W^{(r)}(\tau)$ with consistently increasing virtual queue length over time. With a smaller value of ε, the virtual queue length $W^{(r)}(\tau)$ grows more aggressively with the same number of packet dropping due to the term $\bar{a}_r\,(1 - \varepsilon_r)$ in (4.15), thus imposing more scheduling weight on the urgent packets according to (4.29). In this way, packet urgency plays a more important role in VNF scheduling, resulting in less packet droppings due to expiry.

Fig. 4.7 illustrates the performance comparison between the proposed VNF scheduling algorithm and two benchmark algorithms (without packet rushing) in terms of the individ-

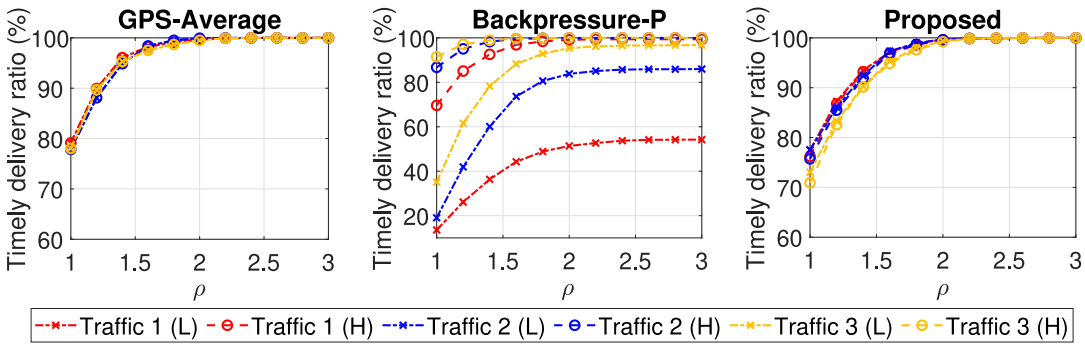
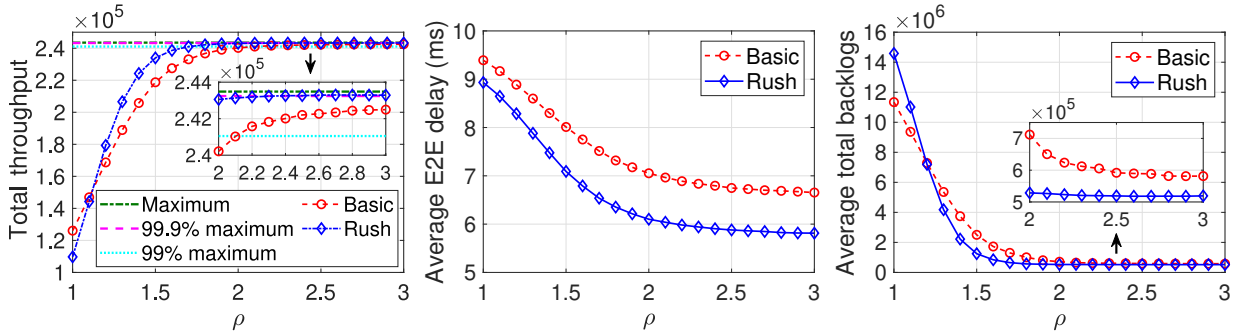


Figure 4.7: Performance comparison between the proposed and benchmark algorithms.

ual timely delivery ratios of different services. We use a simulation setting with topology 2 and traffic set 2, to evaluate the impact of packet arrival rate and processing density on the individual performance of each service with the increase of resource availability. The "GPS-Average" benchmark algorithm corresponds to a GPS resource allocation scheme under the unrealistic assumption of infinitely divisible resources, where VNF $V_h^{(r)}$ enjoys as if a dedicated virtual CPU with a minimum processing rate of $\rho \bar{a}_r P_h^{(r)}$ in cycle per time slot. The virtual CPUs can be scheduled simultaneously at each NFV node, with multiplexing among each other. We see that the timely delivery ratios of services with the same traffic trace and different processing densities overlap with each other, and the timely delivery ratios of services with different traffic traces are close to each other. The "Backpressure-P" benchmark algorithm corresponds to the classical backpressure algorithm adapted for processing resources, in which the differential backlogs in number of required CPU cycles is used as the VNF scheduling weight, and no virtual queues are introduced for individual throughput guarantee. We see a significant performance degradation for low-density and low-rate services. However, the proposed algorithm takes equal importance of the QoS requirement of each service, and achieves similar timely delivery ratios for each service, regardless of the difference in the packet arrival rate and processing density. Moreover, the performance of the proposed algorithm is comparable to that of the "GPS-Average" algorithm, although the proposed algorithm operates in a time-slotted manner with $T = 1ms$ under the constraint that at most one VNF can be scheduled at each NFV node during a time slot.

Next, we evaluate how packet rushing can affect the performance of VNF scheduling. Fig. 4.8 shows comparison of three performance metrics between the basic VNF scheduling



(a) Total throughput (packet/s) (b) Average E2E delay (ms) (c) Average total backlogs

Figure 4.8: Performance comparison without and with packet rushing with the increase of resource availability.

algorithm without packet rushing and the modified VNF scheduling algorithm with packet rushing (denoted by "Basic" and "Rush" respectively), including the total throughput (in packet/s) of all services, the average E2E delay of different services, and the average total backlogs. We see an improvement in all the three performance metrics with the increase of resource availability (indicated by ρ) in both cases without and with packet rushing. As illustrated in Fig. 4.8(a), more packets are timely delivered to the egress edge switch within E2E deadline by taking advantage of packet rushing when there are sufficient resources in the network, i.e., when ρ is greater than a certain value around 1.2. The total throughput achieved by the basic and modified algorithms approach 99% and 99.9% of the maximum value at a resource overprovisioning ratio around 2.1, respectively. However, the strict QoS requirement with $\varepsilon = 10^{-3}$ is difficult to be satisfied without packet rushing, even with further increase of ρ beyond 2.1. We also observe that the modified algorithm with packet rushing cannot outperform the basic algorithm in terms of the total throughput when the resources are limited, e.g., $\rho = 1$. The reason is that the resources allocated to the packets with residual lifetime $1 \leq m < H_r - h + 1$ at VNF $V_h^{(r)}$ have high chances to be eventually wasted due to limited packet rushing opportunity at low resource availability, since such packets can be successfully delivered only by taking advantage of packet rushing. Fig. 4.8(b) shows that the average E2E delay of the services is reduced with packet rushing. With the increase of ρ, the gap between the average E2E delay achieved by the basic and modified algorithms increases to around $1ms$. Fig. 4.8(c) shows that the average total backlogs are reduced with more resources, and the reduction is more significant with packet rushing.

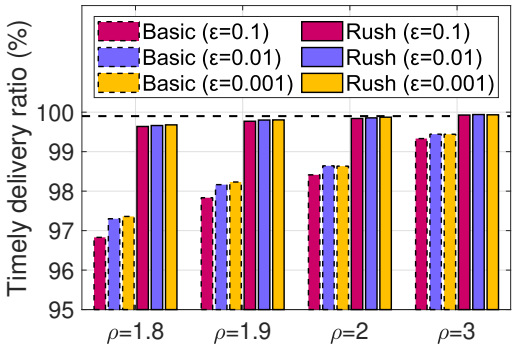
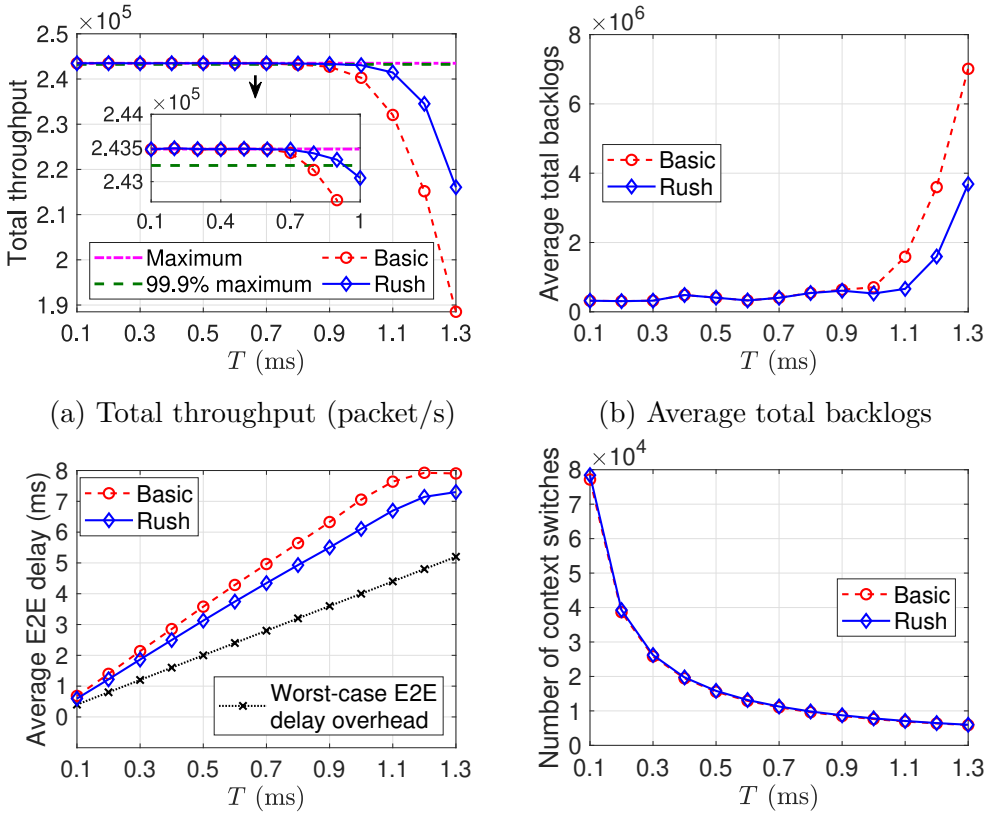


Figure 4.9: Average timely delivery ratio with different QoS constraints.

With the increase of ρ, the QoS performance gradually approaches the QoS requirement as illustrated in Fig. 4.8(a), resulting in a reduced congestion level in the virtual queues. Since packet rushing enhances the QoS performance if resources are sufficient, the virtual queues become even less congested with packet rushing. The physical packet processing queues also become less congested with packet rushing, since the packets can reach the egress edge switch faster on average.

Fig. 4.9 shows the average timely delivery ratio of different services with different QoS constraints ($\varepsilon = 10^{-1}, 10^{-2}, 10^{-3}$) at given values of ρ for both cases without and with packet rushing. With the same QoS constraint, we observe an improvement in the average timely delivery ratio with the increase of ρ and with packet rushing, which is consistent with the results shown in Fig. 4.8. With packet rushing, the difference between the achieved average timely delivery ratios with different QoS constraints at a certain value of ρ is less significant, since a large portion of urgent packets that should have been dropped under the worst-case assumption of no packet rushing can rush to the egress edge switch before expiry, and the achieved average timely delivery ratio approaches 1 at the given values of ρ regardless of the QoS constraint.

To see the impact of time slot length (T) on VNF scheduling performance, we evaluate four performance metrics (including the total throughput, the average total backlogs, the average E2E delay, and the number of context switches per second) at $\rho = 2$, by increasing the time slot length T from $0.1ms$ to $1.3ms$. As illustrated in Fig. 4.10(a), when the algorithm operates with an extremely small time slot length, e.g., $0.1ms$, almost no packets are dropped due to E2E delay violation, resulting in a total throughput approaching 100%



(a) Total throughput (packet/s)

(b) Average total backlogs

(c) Average E2E delay (ms)

(d) No. of context switches per second

Figure 4.10: Performance of the proposed algorithm at different time slot length T ($\rho = 2$).

of the maximum value. The total throughput remains high until T increases to around $0.7ms$, and then degrades significantly with further increase of T. Even if packet rushing is allowed, the QoS violation is significant if T is too large. Correspondingly, the average total backlogs first increase very slowly and then increase sharply due to significant QoS degradation with the increase of T, as illustrated in Fig. 4.10(b). With the increase of T, the worst-case E2E delay overhead, i.e., $H_r T$ for service $r \in \mathcal{R}$, becomes more significant, which cannot be fully compensated by the delay reduction benefit from packet rushing, resulting in an almost linear increasing trend in the average E2E delay for both cases without and with packet rushing, as illustrated in Fig. 4.10(c). The former three performance metrics are all improved with finer granularity of time slot length, at a cost of more switching overhead per second which is nearly inversely proportional to T, as illustrated in Fig. 4.10(d). Hence, the time slot length should not be too small to avoid significant switching overhead. Moreover, if T is too small, the assumption of negligible transmission and propagation delay over the virtual links between consecutive VNFs is non-realistic, and the complexity of the VNF scheduling algorithm grows due to increased size of the packet residual lifetime set at VNF $V_h^{(r)}$, i.e., $|\mathcal{M}_h^{(r)}|$ or $|\tilde{\mathcal{M}}_h^{(r)}|$.

4.6 Summary

In this chapter, we study a delay-aware VNF scheduling problem for deadline-constrained services in a network slicing scenario, to achieve utility maximization in the presence of small-timescale traffic dynamics, while satisfying the throughput requirement of each service. An online distributed VNF scheduling algorithm is proposed for both without and with packet rushing, based on a delay-aware virtual queueing model. The differential backlogs, throughput performance and packet urgency are taken into consideration for VNF scheduling. Simulation results demonstrate an $\left[\mathcal{O}(\frac{1}{\vartheta}), \mathcal{O}(\vartheta)\right]$ utility-backlog trade-off with utility importance parameter ϑ. The effectiveness of virtual queues is verified through a significant QoS performance gap between the proposed algorithm and the "Backpressure-P" benchmark. Even though the proposed algorithm operates in a time granularity of $100\mu s$ to ms, the performance gap with a GPS scheme under the assumption of infinitely divisible resources is not significant. A performance improvement is observed with packet rushing at a sufficiently high resource availability especially in terms of average E2E delay, and the impact of time slot length on VNF scheduling performance is also evaluated.

Chapter 5

Conclusions and Future Research Directions

5.1 Conclusions

The objective of this PhD research is to develop a dynamic resource management framework for embedded services in an SDN/NFV-enabled core network, with the consideration of the unique properties of CPU processing resources in a virtualized network environment, to achieve consistent QoS performance in terms of E2E delay satisfaction and throughput guarantee, by adapting to the network traffic dynamics in different time granularities.

We first develop a delay-aware flow migration model for embedded services with average E2E delay requirements based on a simplified Poisson traffic assumption. The average packet arrival rate of the Poisson traffic is assumed stable within a time interval and varies across different time intervals. A mixed integer optimization problem is formulated, to balance between two conflicting objectives of load balancing and reconfiguration overhead reduction, under processing resource capacity constraints, average E2E delay constraints, and maximal tolerable service downtime constraints. Two solutions are proposed for the optimization problem, including an optimal MIQCP solution and a low-complexity heuristic solution. With flow migration, more traffic from the services can be accommodated with average E2E delay guarantee. Numerical results demonstrate that the proposed flow migration model achieves medium level load balancing without a significant compromise on

reconfiguration overhead. The heuristic solution achieves performance comparable with the optimal MIQCP solution in terms of total cost minimization, with significant improvement on time efficiency.

In the second research problem, we remove the oversimplified Poisson traffic assumption, and investigate when and how to trigger resource scaling and possible flow migrations in a local network segment. A more strict probabilistic delay requirement is considered in developing a QoS-aware resource demand prediction scheme. We use traffic samples of a real-world nonstationary traffic trace in both a medium timescale ($20s$ in simulation) and a smaller timescale ($100ms$ in simulation) for resource demand prediction, based on a change point detection scheme and a GPR-based fBm traffic parameter learning scheme. Packet-level simulations demonstrate the effectiveness of the resource demand prediction scheme in terms of capturing the traffic burstiness in timescales larger than $100ms$. QoS satisfaction is observed for a synthesized packet arrival trace with traffic burstiness only in timescales larger than $100ms$, while occasional QoS violation is observed for the real-world packet arrival trace with traffic burstiness in even smaller time granularities such as $1ms$, especially for the more stringent QoS requirements. The outputs of resource demand prediction, including both the detected change points in time and the predicted resource demands, are applied to a dynamic VNF migration learning module based on a proposed penalty-aware deep $\mathcal{Q}$-learning algorithm. Through reinforcement learning, the patterns in the traffic trace can be captured and VNF migration decisions can be made adaptively to achieve a trade-off among load balancing, migration cost reduction, and resource overloading penalty suppression in the long run. The decision epoch length is time-varying, corresponding to the time duration of each detected stationary traffic segment. Numerical results show that the proposed deep $\mathcal{Q}$-learning algorithm achieves more training loss reduction and more penalty suppression compared with the benchmarks.

In the third research problem, we focus on a sufficiently long time duration with given VNF placement and stationary traffic statistics, and investigate a delay-aware VNF scheduling problem for deadline-constrained services with strict throughput requirements, to achieve total network utility maximization with traffic dynamics in even smaller time granularities (e.g., $100\mu s$-$1ms$). To incorporate packet delay awareness, we use a delay-aware virtual packet processing queueing model. We also replace service throughput requirements by equivalent virtual queue stability requirements for each service. Based on Lyapunov optimization, an online distributed VNF scheduling algorithm is derived, which greedily minimizes a Lyapunov drift-plus-penalty in each time slot. The proposed algorithm

can be executed in a time slotted manner with a realistic *state-of-the-art* time slot length (e.g., $100\mu s$-$1ms$) for CPU resource scheduling. At each NFV node, a VNF with the maximum VNF scheduling weight is scheduled. The scheduling weight for a VNF incorporates the weighted differential backlogs with the downstream VNF, the virtual queue lengths indicating the current throughput performance, and the number of urgent and non-urgent packets. The proposed algorithm achieves asymptotically optimal total network utility with the increase of utility importance parameter, at the cost of linearly increasing average total backlogs. The effectiveness of virtual queues is verified through a significant throughput improvement achieved by the proposed algorithm compared with the "Backpressure-P" benchmark. Also, the proposed time-slotted VNF scheduling algorithm achieves a comparable performance with a GPS benchmark scheme under the unrealistic assumption of infinitely divisible CPU processing resources. To enhance resource efficiency, we consider that a packet can be processed at multiple VNFs within one time slot, which is referred to as packet rushing. A packet processed by more than one VNF in a time slot is referred to as a rushing packet. We propose a modified VNF scheduling algorithm with packet rushing, and derive a packet rushing analysis to analyze the number of rushing packets that can be additionally processed at each VNF if packet rushing is allowed, based on which the true queue length updates can be corrected. With packet rushing, we observe performance improvements especially in the average E2E delay.

5.2 Future Research Directions

This PhD research can be extended in several directions.

In the proposed learning-based VNF migration scheme, the QoS provisioning and the cost minimization are addressed separately. The QoS provisioning is achieved by first predicting a QoS-aware resource demand and then allocating resources accordingly. The cost minimization is achieved by deep $\mathcal{Q}$-learning based VNF migration decision. However, due to the traffic burstiness in finer time granularities than the small time interval (e.g., $100\mu s$) under consideration in the resource demand prediction framework, QoS violation occasionally happens especially for a very stringent QoS requirement. To provide better QoS provisioning, a potential approach is to explicitly incorporate the measured QoS performance metric as a feedback in the MDP state and make decisions accordingly to achieve QoS satisfaction in the long run. In this case, a new MDP formulation with both resource

capacity constraints and QoS constraints should be investigated. The simplest way to deal with constrained MDP is reward shaping, i.e., creating a new reward as a weighted combination of the original reward and the constraint violation penalties, as done in the proposed penalty-aware deep $\mathcal{Q}$-learning approach. However, the weights in the shaped reward are typically difficult to adjust during the training process especially when the MDP model is complex with a high-dimensional state and action space and when there are multiple constraints. A potential approach to deal with constrained MDP is to apply the primal-dual method in RL, in which the weights of the constraint violation penalties are dual variables which can be updated and learned together with the VNF migration policy which corresponds to primal variables.

The VNF migration learning agent is developed for a single VNF in a local neighborhood, in which the dynamics of other VNFs are treated as background traffic at the NFV nodes. In a multi-service scenario, we should have multiple such learning agents working independently at each VNF belonging to different services, which may produce suboptimal VNF migration decisions in terms of an overall cost. A direct extension of the MDP formulation with multiple VNFs may cause the curse of dimensionality issue. Multi-agent or distributed RL is a potential approach to simultaneously making decisions at multiple VNFs towards a common objective, while keeping the state and action space at each learning agent small. Considering multiple VNFs will facilitate exploiting the multiplexing gain among VNFs and improve resource utilization.

The dynamic processing (computing) resource management solutions developed in this PhD research for dealing with traffic dynamics in core networks can be extended to radio access networks with mobile edge computing. The coordination among different types of resources (e.g., communication, computing, and caching resources) at the mobile edge should be investigated, and new issues in the wireless networks should be addressed, such as mobility, channel dynamics, and interference.

References

[1] I. Vision, "Framework and overall objectives of the future development of IMT for 2020 and beyond," *International Telecommunication Union (ITU), Document, Radiocommunication Study Groups*, 2015.

[2] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[3] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.

[4] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in *Proc. IEEE NOMS*, May 2014, pp. 1–8.

[5] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101–109, July 2017.

[6] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Sept. 2016.

[7] V.-G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1567–1602, third quarter 2017.

[8] Q. Duan, N. Ansari, and M. Toy, "Software-defined network virtualization: an architectural framework for integrating SDN and NFV for service provisioning in future networks," *IEEE Netw.*, vol. 30, no. 5, pp. 10–16, Sept. 2016.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[10] C. Lorenz, D. Hock, J. Scherer, R. Durner, W. Kellerer, S. Gebert, N. Gray, T. Zinner, and P. Tran-Gia, "An SDN/NFV-enabled enterprise network architecture offering fine-grained security policy enforcement," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 217–223, Mar. 2017.

[11] ETSI NFV ISG, "NFV-EVE005: SDN usage in NFV architectural framework," Oct. 2015.

[12] Q. Ye, J. Li, K. Qu, W. Zhuang, X. Shen, and X. Li, "End-to-end quality of service in 5G networks – Examining the effectiveness of a network slicing framework," *IEEE Veh. Technol. Mag.*, vol. 13, no. 2, pp. 65–74, June 2018.

[13] O. Alhussein, P. T. Do, Q. Ye, J. Li, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "A virtual network customization framework for multicast services in NFV-enabled core networks," *IEEE J. Select. Areas Commun.*, vol. 38, no. 6, pp. 1025–1039, June 2020.

[14] X. Chen, W. Ni, I. B. Collings, X. Wang, and S. Xu, "Automated function placement and online optimization of network functions virtualization," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1225–1237, Feb. 2019.

[15] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Efficient NFV-enabled multicasting in SDNs," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2052–2070, Mar. 2019.

[16] D. Li, P. Hong, K. Xue *et al.*, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, 2018.

[17] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet of Things J.*, vol. 6, no. 1, pp. 692–704, Feb. 2019.

[18] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. IEEE INFOCOM'18*, Apr. 2018, pp. 486–494.

[19] Z. Luo, C. Wu, Z. Li, and W. Zhou, "Scaling geo-distributed network function chains: A prediction and learning framework," *IEEE J. Select. Areas Commun.*, vol. 37, no. 8, pp. 1838–1850, Aug. 2019.

[20] "MAWI Working Group Traffic Archive," http://mawi.wide.ad.jp/mawi/, 2020, [Online; accessed 17-December-2020].

[21] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, June 2000, pp. 519–528.

[22] J. Rexford, "Route optimization in IP networks," *Handbook of Optimization in Telecommunications*, pp. 679–700, 2006.

[23] H. Tang, D. Zhou, and D. Chen, "Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 530–543, Mar. 2019.

[24] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling innovation in network function control," in *Proc. ACM SIGCOMM*, Aug. 2014, pp. 163–174.

[25] M. Peuster and H. Karl, "E-state: Distributed state management in elastic network function deployments," in *Proc. NetSoft*, June 2016, pp. 6–10.

[26] L. Nobach, I. Rimac, V. Hilt, and D. Hausheer, "Statelet-based efficient and seamless NFV state transfer," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 964–977, Dec. 2017.

[27] B. Zhang, P. Zhang, Y. Zhao, Y. Wang, X. Luo, and Y. Jin, "Co-scaler: Cooperative scaling of software-defined NFV service function chain," in *Proc. IEEE Conf. Network Function Virtualization and Software Defined Networks*, Nov. 2016, pp. 33–38.

[28] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *Proc. IEEE CloudNet*, Oct. 2015, pp. 255–260.

[29] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 543–553, Sept. 2017.

[30] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 343–356, June 2017.

[31] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008–2025, Aug. 2017.

[32] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," in *Proc. IEEE ICNP*, Nov. 2016, pp. 1–10.

[33] J. Xia, D. Pang, Z. Cai, M. Xu, and G. Hu, "Reasonably migrating virtual machine in NFV-featured networks," in *Proc. IEEE Conf. Computer and Information Technology*, Dec. 2016, pp. 361–366.

[34] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206–1243, second quarter 2018.

[35] S. Dräxler, H. Karl, and Z. Á. Mann, "JASPER: Joint optimization of scaling, placement, and routing of virtual network services," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 946–960, June 2018.

[36] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, July 2013.

[37] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," University of Cambridge, Cambridge, U.K., Tech. Rep., 2007.

[38] G. Comert and A. Bezuglov, "An online change-point-based model for traffic parameter prediction," *IEEE Trans. Intell. Transport. Syst.*, vol. 14, no. 3, pp. 1360–1369, Sep. 2013.

[39] C. Fraleigh, F. Tobagi, and C. Diot, "Provisioning IP backbone networks to support latency sensitive traffic," in *Proc. IEEE INFOCOM'03*, Apr. 2003, pp. 1871–1879.

[40] J. Kim and G. Hwang, "Adaptive bandwidth allocation based on sample path prediction with Gaussian process regression," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4983–4996, Oct. 2019.

[41] Y. Cheng, W. Zhuang, and L. Wang, "Calculation of loss probability in a finite size partitioned buffer for quantitative assured service," *IEEE Trans. Commun.*, vol. 55, no. 9, pp. 1757–1771, Aug. 2007.

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2011.

[43] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti, "Cellular network traffic scheduling with deep reinforcement learning," in *Proc. AAAI'18*, Feb. 2018.

[44] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, to appear, doi: 10.1109/TETC.2019.2902661.

[45] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of things with edge computing," *IEEE netw.*, vol. 32, no. 1, pp. 96–101, Jan.-Feb. 2018.

[46] J. Li, W. Shi, N. Zhang, and X. Shen, "Delay-aware VNF scheduling: A reinforcement learning approach with variable action set," *IEEE Trans. Cogn. Commun. Netw.*, 2020, to appear, doi: 10.1109/TCCN.2020.2988908.

[47] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks," *IEEE J. Select. Areas Commun.*, vol. 38, no. 1, pp. 217–228, Jan. 2020.

[48] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 692–704, Feb. 2019.

[49] P. Emmerich, D. Raumer, S. Gallenmüller, F. Wohlfart, and G. Carle, "Throughput and latency of virtual switching with Open vSwitch: A quantitative analysis," *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 314–338, Apr. 2018.

[50] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. Ramakrishnan, T. Wood *et al.*, "NFVnice: Dynamic backpressure and scheduling for NFV service chains," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 639–652, Apr. 2020.

[51] S. R. Chowdhury, T. Bai, R. Boutaba, J. François *et al.*, "UNiS: A user-space non-intrusive workflow-aware virtual network function scheduler," in *2018 14th International Conf. on Network and Service Management (CNSM)*, 2018, pp. 152–160.

[52] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE netw.*, vol. 34, no. 3, pp. 134–142, May/June 2020.

[53] H. Deng, T. Zhao, and I.-H. Hou, "Online routing and scheduling with capacity redundancy for timely delivery guarantees in multihop networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1258–1271, June 2019.

[54] R. Singh and P. Kumar, "Throughput optimal decentralized scheduling of multihop networks with end-to-end deadline constraints: Unreliable links," *IEEE Trans. Automat. Contr.*, vol. 64, no. 1, pp. 127–142, Jan. 2019.

[55] R. Li and A. Eryilmaz, "Scheduling for end-to-end deadline-constrained traffic with reliability requirements in multihop networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1649–1662, Oct. 2012.

[56] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, 2016.

[57] S. Yang, F. Li, R. Yahyapour, and X. Fu, "Delay-sensitive and availability-aware virtual network function scheduling for NFV," *IEEE Trans. Serv. Comput.*, to appear, doi: 10.1109/TSC.2019.2927339.

[58] Y. Zhang, F. He, T. Sato, and E. Oki, "Network service scheduling with resource sharing and preemption," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 764–778, June 2020.

[59] H. Alameddine, M. H. K. Tushar, and C. Assi, "Scheduling of low latency services in softwarized networks," *IEEE Trans. Cloud Comput.*, to appear, doi: 10.1109/TCC.2019.2907949.

[60] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. IEEE INFOCOM'09*, 2009, pp. 2936–2940.

[61] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2118–2131, Oct. 2018.

[62] L. Gu, D. Zeng, S. Tao, S. Guo, H. Jin, A. Y. Zomaya, and W. Zhuang, "Fairness-aware dynamic rate control and flow scheduling for network utility maximization in network service chain," *IEEE J. Select. Areas Commun.*, vol. 37, no. 5, pp. 1059–1071, May 2019.

[63] "Gurobi Optimizer Reference Manual," https://www.gurobi.com/documentation/9.1/refman/constraints.html, 2018, [Online; accessed 17-December-2020].

[64] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Dynamic flow migration for embedded services in SDN/NFV-enabled 5G core networks," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2394–2408, Apr. 2020.

[65] ——, "Traffic engineering for service-oriented 5G networks with SDN-NFV integration," *IEEE Netw.*, vol. 34, no. 4, pp. 234–241, July/Aug. 2020.

[66] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Delay-aware flow migration for embedded services in 5G core networks," in *Proc. IEEE ICC*, May 2019, pp. 1–6.

[67] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Dynamic resource scaling for VNF over nonstationary traffic: A learning approach," *IEEE Trans. Cogn. Commun. Netw.*, to appear, doi: 10.1109/TCCN.2020.3018157.

[68] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep 2017.

[69] M. Shin, S. Chong, and I. Rhee, "Dual-resource TCP/AQM for processing-constrained networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 435–449, Apr. 2008.

[70] L. Rizzo, M. Carbone, and G. Catalli, "Transparent acceleration of software packet forwarding using Netmap," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2471–2479.

[71] S. Garzarella, G. Lettieri, and L. Rizzo, "Virtual device passthrough for high speed VM networking," in *Proc. 11th ACM/IEEE Symp. Architectures for Networking and Communications Systems*, May 2015, pp. 99–110.

[72] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet of Things J.*, vol. 6, no. 1, pp. 692–704, Feb. 2019.

[73] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*, vol. 102, pp. 1–16, Apr. 2017.

[74] F. Ben Jemaa, G. Pujolle, and M. Pariente, "Analytical models for QoS-driven VNF placement and provisioning in wireless carrier cloud," in *Proc. 19th ACM International Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Nov. 2016, pp. 148–155.

[75] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: wh freeman, 1978.

[76] B. Krithikaivasan, Y. Zeng, K. Deka, and D. Medhi, "ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 683–696, June 2007.

[77] A. Bayati, V. Asghari, K. Nguyen, and M. Cheriet, "Gaussian process regression based traffic modeling and prediction in high-speed networks," in *Proc. IEEE GLOBECOM'16*, Dec. 2016, pp. 1–7.

[78] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[79] H. S. Kim and N. B. Shroff, "Loss probability calculations and asymptotic analysis for finite buffer multiplexers," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 755–768, Dec. 2001.

[80] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[81] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE INFOCOM'18*, Apr. 2018, pp. 1871–1879.

[82] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR'16*, May 2016, pp. 1–7.

[83] P. Abry and F. Sellan, "The wavelet-based synthesis for fractional Brownian motion proposed by F. Sellan and Y. Meyer: Remarks and fast implementation," *Applied and Computational Harmonic Analysis*, vol. 3, no. 4, pp. 377–383, Oct. 1996.

[84] H. Kobayashi and B. L. Mark, *System Modeling and Analysis: Foundations of System Performance Evaluation*. Pearson Education India, 2009.

[85] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[86] P. Lu, Q. Sun, K. Wu, and Z. Zhu, "Distributed online hybrid cloud management for profit-driven multimedia cloud computing," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1297–1308, 2015.

[87] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin, "Carbon-aware online control of geo-distributed cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2506–2519, 2015.

[88] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang *et al.*, "Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization," *IEEE Trans. Service Comput.*, vol. 8, no. 3, pp. 398–409, 2015.

[89] K. P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," University of British Columbia, Vancouver, Canada, Tech. Rep., 2007.

[90] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, U.S.A.: Springer, 2006.

Appendix A

Derivation of $\tilde{\alpha}_n(k)$

Let $\eta_n^{rh}(k)$ denote a ratio between resources occupied by VNF $V_h^{(r)}$ and resource capacity of NFV node n, given by

$$\eta_n^{rh}(k) = \frac{P_n^{rh} b_r}{\mathsf{C}_n} \left(\lambda^{(r)}(k) + \frac{1}{D^{rh}(k)} \right) \mathrm{x}_n^{rh}(k). \tag{A1}$$

VNF set $\mathcal{V}$ is divided into two subsets, i.e., $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where $\mathcal{V}_1 = \{(r,h) \in \mathcal{V} | \mathrm{x}_n^{rh}(k) = f_2^{(r)} = 1\}$ is a set of VNFs belonging to SFC category III on NFV node n, and $\mathcal{V}_2$ is a set of all other VNFs. Vertical delay scaling is applied to only VNFs in $\mathcal{V}_1$. Before delay scaling, resource usage at NFV node n is composed of three parts, given by

$$\eta_n(k) = \sum_{(r,h) \in \mathcal{V}_1 \cup \mathcal{V}_2} \eta_n^{rh}(k) + \sum_{(r,h) \in \mathcal{V}_1 \cup \mathcal{V}_2} \frac{\mathrm{w}_n(k) \mathrm{x}_n^{rh}(k) \mathtt{W}}{T_n}. \tag{A2}$$

The ratio of resources occupied by VNFs in $\mathcal{V}_1$ before vertical delay scaling is given by

$$\sum_{(r,h) \in \mathcal{V}_1} \eta_n^{rh}(k) = \sum_{(r,h) \in \mathcal{V}_1} \frac{P_n^{rh} b_r}{\mathsf{C}_n} \left(\lambda^{(r)}(k) + \frac{1}{\mathsf{D}^{rh}(k)} \right). \tag{A3}$$

For a vertical delay scaling by a positive coefficient $\tilde{\alpha}_n(k)$ to increase loading factor of NFV node n from $\eta_n(k)$ to η_{th}, we have the following relationship among parameters, given by

$$\eta_{th} - \sum_{(r,h) \in \mathcal{V}_2} \eta_n^{rh}(k) - \sum_{(r,h) \in \mathcal{V}_1 \cup \mathcal{V}_2} \frac{\mathrm{w}_n(k) \mathrm{x}_n^{rh}(k) \mathtt{W}}{T_n} = \sum_{(r,h) \in \mathcal{V}_1} \frac{P_n^{rh} b_r}{\mathsf{C}_n} \left(\lambda^{(r)}(k) + \frac{1}{\tilde{\alpha}_n(k) \mathsf{D}^{rh}(k)} \right). \tag{A4}$$

Subtracting (A3) from (A4) and arranging items, we obtain

$$\tilde{\alpha}_n(k) = \frac{\sum_{(r,h)\in\mathcal{V}_1} \frac{P_n^{rh} b_r}{\mathsf{D}^{rh}(k)}}{[\eta_{th} - \eta_n(k)]\, \mathsf{C}_n + \sum_{(r,h)\in\mathcal{V}_1} \frac{P_n^{rh} b_r}{\mathsf{D}^{rh}(k)}} \tag{A5}$$

which is equivalent to (2.25).

Appendix B

Bayesian Online Change Point Detection

Under the assumption of *a priori* geometric inter-arrivals of change points, the conditional prior probability distribution over the run length, denoted by $\mathbb{P}(l_i|l_{i-1})$, is given by

$$\mathbb{P}(l_i|l_{i-1}) = \begin{cases} 1-(1/\bar{l}), & \text{if } l_i = l_{i-1}+1 \\ 1/\bar{l}, & \text{if } l_i = 0 \\ 0, & \text{otherwise} \end{cases} \tag{B1}$$

where $\bar{l}$ is the prior average run length. We set $\bar{l} = \frac{3600}{T_M}$ in simulation, corresponding to one hour in time duration. The conditional prior has nonzero mass at only two outcomes, i.e., the run length either grows by 1, or resets to 0. Accordingly, there are two branches for $\mathbb{P}(l_i, \boldsymbol{x}_i)$, given by

$$\overbrace{\mathbb{P}(l_i, \boldsymbol{x}_i)}^{i\text{-th iteration}} = \begin{cases} \left(1-\frac{1}{\bar{l}}\right)\mathbb{P}\left(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)}\right)\,\mathbb{P}\left(l_{i-1}, \boldsymbol{x}_{i-1}\right), & \text{if } l_i = l_{i-1}+1 \\ \frac{1}{\bar{l}}\sum_{l_{i-1}=0}^{i-1}\mathbb{P}\left(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)}\right)\,\mathbb{P}\left(l_{i-1}, \boldsymbol{x}_{i-1}\right), & \text{if } l_i = 0. \end{cases} \tag{B2}$$

The predictive model $\mathbb{P}(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)})$ evaluates the probability that x_i belongs to the same run as $\boldsymbol{x}_{i-1}^{(l)}$, given run length l_{i-1}. Under the i.i.d Gaussian assumption with unknown mean μ and variance ν^2, a Normal-Inverse-Gamma prior is placed on μ and ν^2, given by

$$\mathbb{P}(\mu, \nu^2) \sim \mathcal{N}\left(\mu|\mu_0, \frac{\nu^2}{\kappa_0}\right)\mathcal{IG}\left(\nu^2|\alpha_0, \beta_0\right) \tag{B3}$$

where $\{\mu_0, \kappa_0, \alpha_0, \beta_0\}$ are prior parameters. Conjugate Bayesian analysis [89, 90] gives a Normal-Inverse-Gamma posterior on μ and ν^2 given $\boldsymbol{x}_{i-1}^{(l)}$, represented as

$$\mathbb{P}(\mu,\nu^2|\boldsymbol{x}_{i-1}^{(L)}) \sim \mathcal{N}\left(\mu|\mu_{i-1}^{(L)}, \frac{\nu^2}{\kappa_{i-1}^{(L)}}\right)\mathcal{IG}\left(\nu^2|\alpha_{i-1}^{(L)}, \beta_{i-1}^{(L)}\right) \tag{B4}$$

where $\{\mu_{i-1}^{(l)}, \kappa_{i-1}^{(l)}, \alpha_{i-1}^{(l)}, \beta_{i-1}^{(l)}\}$ are referred to as sufficient statistics corresponding to $\boldsymbol{x}_{i-1}^{(l)}$. Each possible value of run length l_{i-1} corresponds to a group of sufficient statistics. The posterior predictive distribution for x_i given l_{i-1} and $\boldsymbol{x}_{i-1}^{(l)}$, i.e., $\mathbb{P}(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)})$, is described by a student-t distribution, represented as

$$\mathbb{P}(x_i|l_{i-1}, \boldsymbol{x}_{i-1}^{(l)}) \sim t_{2\alpha_{i-1}^{(l)}}\left(x_i|\mu_{i-1}^{(l)}, \frac{\beta_{i-1}^{(l)}(\kappa_{i-1}^{(l)}+1)}{\kappa_{i-1}^{(l)}\alpha_{i-1}^{(l)}}\right) \tag{B5}$$

where $\mu_{i-1}^{(l)}$ is the mean, $2\alpha_{i-1}^{(l)}$ is the degrees of freedom, and $\frac{\beta_{i-1}^{(l)}(\kappa_{i-1}^{(l)}+1)}{\kappa_{i-1}^{(l)}\alpha_{i-1}^{(l)}}$ is the scale. The standard deviation of the student-t distribution, denoted by $\nu_{i-1}^{(l)}$, is given by

$$\nu_{i-1}^{(l)} = \sqrt{\frac{\beta_{i-1}^{(l)}\left(\kappa_{i-1}^{(l)}+1\right)}{\kappa_{i-1}^{(l)}\left(\alpha_{i-1}^{(l)}-1\right)}}. \tag{B6}$$

After new observation x_i is available, sufficient statistics corresponding to $\boldsymbol{x}_i^{(l)}$ for $\forall l_i > 0$ are updated as

$$\mu_i^{(l)} = \frac{\kappa_{i-1}^{(l)}\mu_{i-1}^{(l)} + x_i}{\kappa_{i-1}^{(l)}+1} \tag{B7a}$$

$$\kappa_i^{(l)} = \kappa_{i-1}^{(l)} + 1 \tag{B7b}$$

$$\alpha_i^{(l)} = \alpha_{i-1}^{(l)} + \frac{1}{2} \tag{B7c}$$

$$\beta_i^{(l)} = \beta_{i-1}^{(l)} + \frac{\kappa_{i-1}^{(l)}\left(x_i - \mu_{i-1}^{(l)}\right)^2}{2\left(\kappa_{i-1}^{(l)}+1\right)}. \tag{B7d}$$

For $l_i = 0$, the sufficient statistics are updated from the prior parameters of the Normal-Inverse-Gamma distribution.

Appendix C

Proof of Lemma 1

Let C_{max} be the maximum processing resource budget (in cycle per time slot) among all the NFV nodes in set $\mathcal{N}$, and let P_{min} be the minimum processing density (in cycle/packet) among all VNFs. We have $S^{(r)}_{h,m}(\tau) \leq S^{(r)}_{h}(\tau) \leq \frac{C_{max}}{P_{min}} = S_{max}$, $D^{(r)}_{h}(\tau) \leq Q^{(r)}_{h,\mathbb{U}}(\tau) \leq A^{(r)}_{max}$ and $\chi^{(r)}(\tau) \leq A^{(r)}_{max}$. Based on the inequality $([a-b]^{+}+c)^2 \leq a^2+b^2+c^2-2a(b-c)$ for $\forall a,b,c \geq 0$, we can obtain from the queue update equations (4.1), (4.15) and (4.16) that

$$\begin{aligned}
&\sum_{h\in\mathcal{H}_r}\left[P^{(r)}_h q^{(r)}_h(\tau+1)\right]^2 \leq \sum_{h\in\mathcal{H}_r}\left[P^{(r)}_h q^{(r)}_h(\tau)\right]^2 \\
&-2\sum_{h\in\mathcal{H}_r}(P^{(r)}_h)^2 q^{(r)}_h(\tau)\left[S^{(r)}_h(\tau)-S^{(r)}_{h,\mathbb{U}}(\tau)-S^{(r)}_{h-1}(\tau)\mathbb{1}\{h>1\}-A^{(r)}(\tau)\mathbb{1}\{h=1\}\right] \\
&\underbrace{+\sum_{h\in\mathcal{H}_r}(P^{(r)}_h)^2\left[S^{(r)}_h(\tau)+D^{(r)}_h(\tau)\right]^2+\sum_{h\in\mathcal{H}_r}(P^{(r)}_h)^2\left[S^{(r)}_{h-1}(\tau)\mathbb{1}\{h>1\}+A^{(r)}(\tau)\mathbb{1}\{h=1\}\right]^2}_{\leq \mathsf{B}^{(r)}_1}
\end{aligned} \tag{C1}$$

$$\begin{aligned}
W^{(r)}(\tau+1)^2 \leq\ & W^{(r)}(\tau)^2 - 2W^{(r)}(\tau)\left[A^{(r)}(\tau)+\sum_{h\in\mathcal{H}_r}S^{(r)}_{h,\mathbb{U}}(\tau)-\sum_{h\in\mathcal{H}_r}Q^{(r)}_{h,\mathbb{U}}(\tau)-\bar{a}_r\left(1-\varepsilon_r\right)\right] \\
&\underbrace{+\left[A^{(r)}(\tau)+\sum_{h\in\mathcal{H}_r}S^{(r)}_{h,\mathbb{U}}(\tau)\right]^2+\left[\sum_{h\in\mathcal{H}_r}Q^{(r)}_{h,\mathbb{U}}(\tau)\right]^2+\left(\bar{a}_r\right)^2\left(1-\varepsilon_r\right)^2+2\bar{a}_r\left(1-\varepsilon_r\right)\sum_{h\in\mathcal{H}_r}Q^{(r)}_{h,\mathbb{U}}(\tau)}_{\leq \mathsf{B}^{(r)}_2}
\end{aligned} \tag{C2}$$

$$F^{(r)}(\tau+1)^2 \leq F^{(r)}(\tau)^2 - 2F^{(r)}(\tau)\left[A^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} S^{(r)}_{h,\mathbb{U}}(\tau) - \sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau) - \chi^{(r)}(\tau)\right]$$
$$+ \underbrace{\left[A^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} S^{(r)}_{h,\mathbb{U}}(\tau)\right]^2 + \left[\sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau)\right]^2 + \chi^{(r)}(\tau)^2}_{\leq \mathsf{B}^{(r)}_3} + 2\chi^{(r)}(\tau)\sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau) \tag{C3}$$

for service $r \in \mathcal{R}$, where $\mathsf{B}^{(r)}_1$, $\mathsf{B}^{(r)}_2$ and $\mathsf{B}^{(r)}_3$ are constants given by

$$\begin{aligned}
\mathsf{B}^{(r)}_1 &= (P^{(r)}_{max})^2\left[H_r\left(S_{max} + A^{(r)}_{max}\right)^2 + (H_r - 1)(S_{max})^2 + (A^{(r)}_{max})^2\right] \\
\mathsf{B}^{(r)}_2 &= (A^{(r)}_{max} + H_r S_{max})^2 + (H_r A^{(r)}_{max})^2 + (\bar{a}_r)^2(1-\varepsilon_r)^2 + 2\bar{a}_r(1-\varepsilon_r)H_r A^{(r)}_{max} \\
\mathsf{B}^{(r)}_3 &= (A^{(r)}_{max} + H_r S_{max})^2 + (H_r A^{(r)}_{max})^2 + (A^{(r)}_{max})^2.
\end{aligned} \tag{C4}$$

Using the inequalities in (C1), (C2) and (C3), we can obtain that

$$\begin{aligned}
&\Delta\left(\boldsymbol{\Theta}(\tau)\right) - \vartheta\mathbb{E}\left\{\sum_{r\in\mathcal{R}}\phi(\chi^{(r)}(\tau))|\boldsymbol{\Theta}(\tau)\right\} \leq \sum_{r\in\mathcal{R}}\mathsf{B}_r \\
&- \sum_{r\in\mathcal{R}}\mathbb{E}\left\{\sum_{h\in\mathcal{H}_r}\left(P^{(r)}_h\right)^2 q^{(r)}_h(\tau)\left[S^{(r)}_h(\tau) - S^{(r)}_{h,\mathbb{U}}(\tau) - S^{(r)}_{h-1}(\tau)\mathbb{1}\{h>1\} - A^{(r)}(\tau)\mathbb{1}\{h=1\}\right]|\boldsymbol{\Theta}(\tau)\right\} \\
&- \sum_{r\in\mathcal{R}}\mathbb{E}\left\{\left(\frac{\varphi}{\bar{a}_r}\right)^2 W^{(r)}(\tau)\left[A^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} S^{(r)}_{h,\mathbb{U}}(\tau) - \sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau) - \bar{a}_r(1-\varepsilon_r)\right]|\boldsymbol{\Theta}(\tau)\right\} \\
&- \sum_{r\in\mathcal{R}}\mathbb{E}\left\{(P_r)^2 F^{(r)}(\tau)\left[A^{(r)}(\tau) + \sum_{h\in\mathcal{H}_r} S^{(r)}_{h,\mathbb{U}}(\tau) - \sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau) - \chi^{(r)}(\tau)\right]|\boldsymbol{\Theta}(\tau)\right\} \\
&+ \sum_{r\in\mathcal{R}}\mathbb{E}\left\{(P_r)^2\chi^{(r)}(\tau)\sum_{h\in\mathcal{H}_r} Q^{(r)}_{h,\mathbb{U}}(\tau)|\boldsymbol{\Theta}(\tau)\right\} - \vartheta\mathbb{E}\left\{\sum_{r\in\mathcal{R}}\phi(\chi^{(r)}(\tau))|\boldsymbol{\Theta}(\tau)\right\}
\end{aligned} \tag{C5}$$

where

$$\mathsf{B}_r = \frac{1}{2}\left[\mathsf{B}^{(r)}_1 + \left(\frac{\varphi}{\bar{a}_r}\right)^2\mathsf{B}^{(r)}_2 + (P_r)^2\mathsf{B}^{(r)}_3\right]. \tag{C6}$$

The inequality in (C5) can be rewritten as (4.20) under constraint (4.11).