Merancang *reward function* linear dengan penalti SLA (Service Level Agreement) adalah langkah krusial dalam RL untuk *network slicing*, karena Anda harus menyeimbangkan tujuan yang saling bertentangan: memaksimalkan *throughput* (misal: untuk Port 2 Kamera) sambil meminimalkan *latency* (misal: untuk Port 4 Healthcare).
Berdasarkan literatur yang Anda berikan, berikut adalah pendekatan sistematis untuk merancang fungsi tersebut:

1. Struktur Dasar Linear (Weighted Sum)

Bentuk paling umum dari *reward function* linear dalam manajemen jaringan adalah penjumlahan terbobot dari metrik performa utama. Sebagaimana dijelaskan dalam **Xia et al.** dan **Mao et al.**, format dasarnya adalah:
$$R_t = \alpha \cdot \text{Throughput}_t - \beta \cdot \text{Latency}_t - \gamma \cdot \text{PacketLoss}_t$$

- **α, β, γ**: Bobot (koefisien) yang menentukan prioritas 1, 2.
- **Throughput**: Diberi tanda positif (agen mendapat poin jika throughput tinggi).
- **Latency & Loss**: Diberi tanda negatif (agen dihukum jika nilai ini tinggi).

2. Mekanisme Penalti SLA (SLA Violation Penalty)

Untuk kasus Anda di mana **Port 4 (Healthcare)** memiliki batas kritis (misal: delay harus < 10ms), sekadar mengurangi nilai reward linear seringkali tidak cukup. Anda perlu mekanisme penalti yang tegas. Berikut tiga metode berdasarkan referensi:

A. Penalti Berbasis Regret (Regret-Based)

Metode ini menghitung "penyesalan" atau seberapa jauh performa meleset dari target SLA. Ini sangat efektif untuk menormalkan penalti agar sebanding.Berdasarkan **Zeng**, *regret* untuk latensi dapat dirumuskan sebagai:
$$V_{SLA} = \max\left(\frac{D_{actual} - D_{target}}{D_{target}}, 0 \right)$$

- Jika $D_{actual} \le D_{target}$ (SLA terpenuhi), penaltinya 0.
- Jika melanggar, penalti akan membesar secara linear sesuai tingkat pelanggaran 3, 4.

B. Penalti Probabilistik/Biner (Hard Penalty)

Digunakan dalam **Hierarchical DRL** untuk kasus URLLC. Jika latensi melebihi ambang batas (τ_{SLA}), berikan penalti konstan yang besar.
$$R_{penalty} = \begin{cases} -P & \text{jika } \text{Latency} > \tau_{SLA} \\ 0 & \text{jika } \text{Latency} \le \tau_{SLA} \end{cases}$$
Dimana P adalah nilai negatif besar (misal -100). Metode ini sederhana namun bisa membuat gradien tidak stabil saat training 5.

C. Penalti Sigmoid (Soft/Risk-Sensitive Penalty)

Untuk menghindari ketidakstabilan "hard penalty", **SafeSlice** dan **Raftopoulos et al.** menyarankan penggunaan fungsi Sigmoid. Ini memberikan penalti yang "lembut" saat mendekati batas SLA dan penalti berat saat melanggar, membantu agen PPO belajar lebih halus.
$$R_{penalty} = \frac{1}{1 + e^{-k \cdot (\text{Latency} - \text{Threshold})}}$$

Semakin tinggi nilai k, semakin curam fungsi penaltinya (mendekati *hard constraint*) 6, 7.

3. Usulan Formulasi Reward untuk Paper Anda

Menggabungkan konsep di atas untuk skenario 3 Port Anda, berikut adalah rancangan *reward function* yang direkomendasikan:
$$R_{total} = \underbrace{w_1 \cdot \frac{T_{p2}}{T_{max}}}_{\text{Reward Kamera}} - \underbrace{w_2 \cdot \frac{D_{p4}}{D_{target}}}_{\text{Biaya Latency}} - \underbrace{w_3 \cdot V_{SLA}(p4)}_{\text{Penalti Pelanggaran}}$$
Detail Implementasi:

- **Normalisasi:** Sangat penting menormalkan nilai (misal: bagi throughput dengan kapasitas maksimum link) agar nilai reward berada dalam rentang skala yang sama (misal -1 sampai 1). Tanpa normalisasi, agen akan mengabaikan latensi (ms) karena nilainya kecil dibanding throughput (Mbps) 8, 9.
- **Bobot (w):**
- Berikan w_3 nilai terbesar untuk menegaskan bahwa pelanggaran SLA di Port 4 tidak dapat ditoleransi.
- Penelitian menunjukkan bahwa memberikan penalti *Lagrangian* (mengubah bobot penalti secara dinamis saat pelanggaran sering terjadi) dapat meningkatkan kepatuhan SLA 10, 11.

4. Contoh Pseudo-code (Python Style)

Berdasarkan logika **Reward Clipping** dari referensi 12, 13:

```
def calculate_reward(state, action):
    # Port 2: Camera (Maximize Throughput)
    reward_throughput = state.rx_mbps_p2 / MAX_BANDWIDTH

    # Port 4: Healthcare (Minimize Latency & SLA)
    latency_p4 = state.delay_ms_p4
    SLA_THRESHOLD = 10.0 # ms

    if latency_p4 <= SLA_THRESHOLD:
        # Reward positif kecil jika latency bagus
        reward_latency = 1.0 - (latency_p4 / SLA_THRESHOLD)
        penalty = 0
    else:
        # Penalti linear besar jika melanggar
        # Referensi [13] menyarankan format: -C1 + (diff * C2)
        penalty = -10 * (latency_p4 - SLA_THRESHOLD)
        reward_latency = 0

    # Total Linear Combination
    total_reward = (0.4 * reward_throughput) + (0.6 *
(reward_latency + penalty))

    return total_reward
```

Pendekatan ini memastikan model (DQN/PPO) Anda "takut" melanggar batas 10ms di Port 4, namun tetap "termotivasi" menaikkan bandwidth di Port 2 selama kondisi aman.