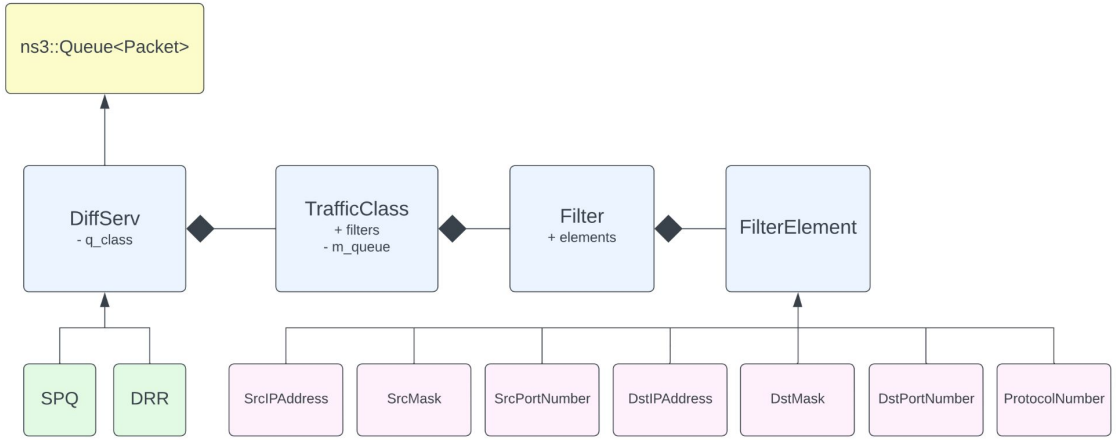
DiffServ

Ashley Radford

Design Overview



DiffServ Details

Schedule
- Pure virtual method
- Peeks at the next scheduled packet
- Returns a const packet, is a const method

Classify
- Uses TrafficClass filters to match packets

AddQueue
- Adds TrafficClass objects to q_class

GetQueues
- So that subclasses can access q_class if needed

```
class DiffServ : public Queue<Packet>
{
   public:
       DiffServ();

       bool Enqueue(Ptr<Packet> p) override;
       Ptr<Packet> Dequeue() override;
       Ptr<Packet> Remove() override;
       Ptr<const Packet> Peek() const override;
       virtual Ptr<const Packet> Schedule() const = 0;
       virtual uint32_t Classify(Ptr<Packet> p);
       virtual void AddQueue(TrafficClass *q);

   protected:
       std::vector<TrafficClass*> GetQueues() const;

   private:
       std::vector<TrafficClass*> q_class;

       bool DoEnqueue(Ptr<Packet> p);
       Ptr<Packet> DoDequeue();
       Ptr<Packet> DoRemove();
       Ptr<const Packet> DoPeek() const;
};
```

DiffServ Methods

Classify() → index

Schedule() → q_class[index].Peek()

Enqueue() → DoEnqueue() → Classify() → q_class[index].Enqueue()

Dequeue() → DoDequeue() → Schedule() → Classify() → q_class[index].Dequeue()

Remove() → DoRemove() → Schedule() → Classify() → q_class[index].Remove()

Peek() → DoPeek() → Schedule()

SPQ Implementation

Overrides:

- Schedule()

Configurations: (see configs)

- QoS Name
- MaxPackets
- Priority
- Default
- DestPort

```
class SPQ : public DiffServ
{
    public:
        SPQ();

        Ptr<const Packet> Schedule() const override;
};
```

DRR Implementation

Overrides:

- Dequeue()
  - Update active_queue
  - Update deficit_counter
- Schedule()
- AddQueue()
  - Add new index to deficit_counter vector

Configurations: (see configs)

- QoS Name
- MaxPackets
- Weight
- Default
- DestPort

```
class DRR : public DiffServ
{
  public:
      DRR();

      Ptr<Packet> Dequeue() override;
      Ptr<const Packet> Schedule() const override;
      void AddQueue(TrafficClass *q) override;

  private:
      uint32_t active_queue;
      mutable uint32_t next_active_queue;
      std::vector<uint32_t> deficit_counter;
      mutable std::vector<uint32_t> next_deficit_counter;
};
```

Design Challenges and Suggestions

Challenge: implementation of Schedule

- After finding the next scheduled queue, do we:
  - Return the dequeued packet
  - Return the peeked at packet

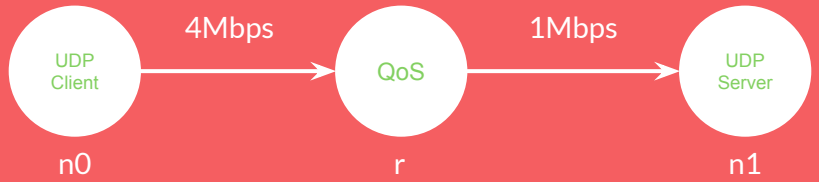
Suggestion: change Schedule's return type

- Index of the next queue to be served
  - No redundant classification needed
  - Lessens the const implementation complexities
- Would still require one of ns3's queue method overrides

Network Topology (see and run simulation)

SPQ Simulation
UDP Client App 1 → 7000
UDP Client App 2 → 9000

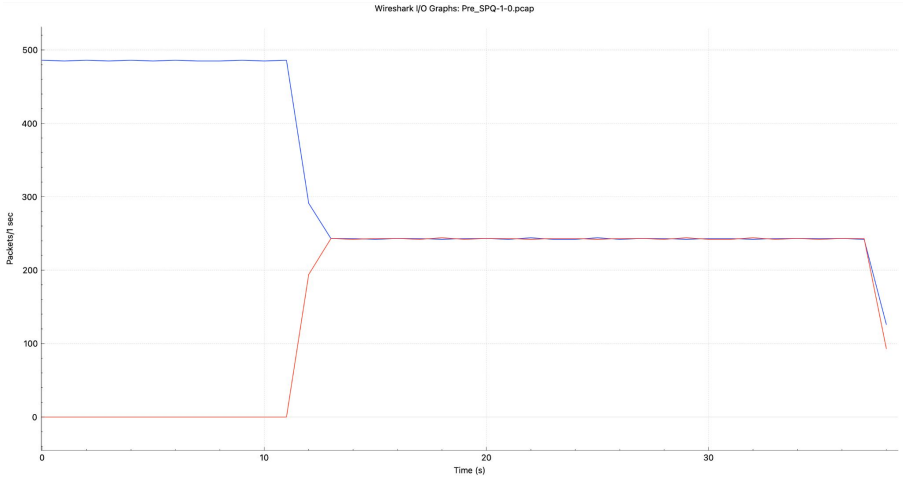
SPQ Simulation
Port 7000: Priority 2
Port 9000: Priority 1



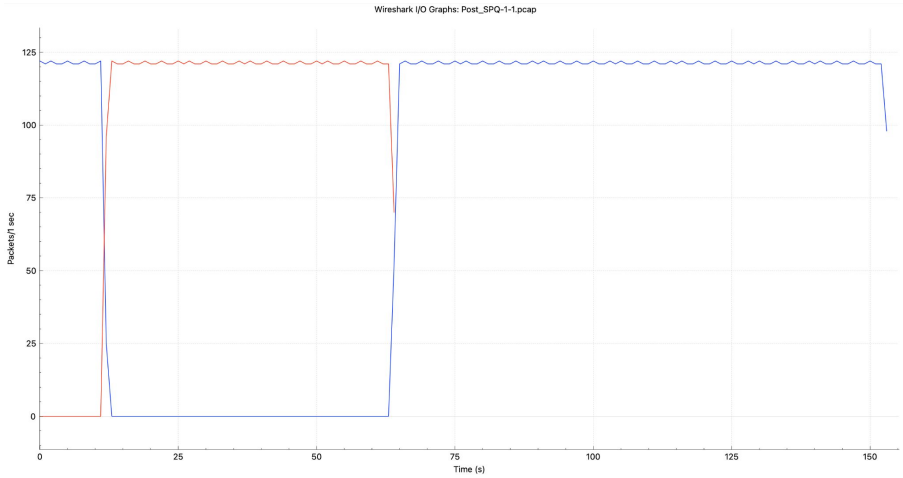
DRR Simulation
UDP Client App 1 → 6000
UDP Client App 2 → 7000
UDP Client App 3 → 9000

DRR Simulation
Port 6000: Weight 100
Port 7000: Weight 200
Port 9000: Weight 300

SPQ Validation

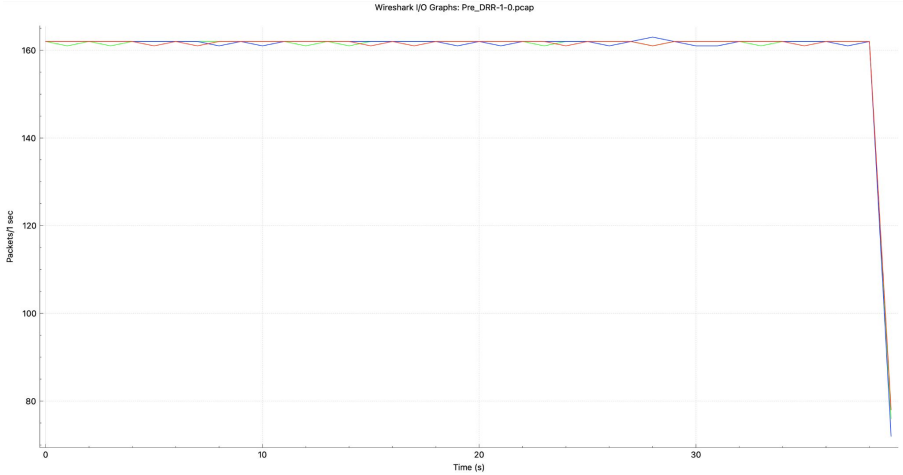


Pre SPQ

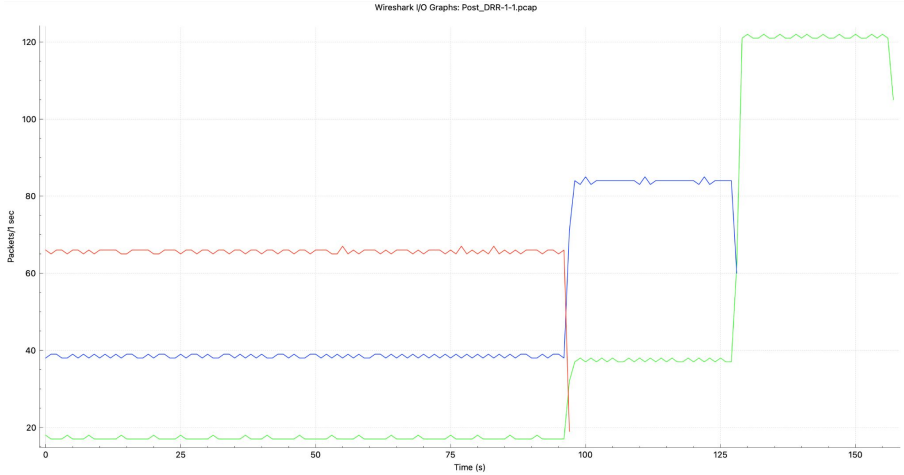


Post SPQ

DRR Validation



Pre DRR



Post DRR

Thank you