Asynchronous Deterministic Networking (ADN) Framework for Large scale networks

draft-joung-detnet-asynch-detnet-framework-00

Jinoo Joung, Jeong-dong Ryoo, Tae-sik Cheung, Yizhou Li, Peng Liu

IETF 114

Scope

- It specifies the framework for both latency & jitter bounds guarantee in large scale networks with dynamic sources with arbitrary input patterns.
  - large scale:
    - arbitrary topology, may include loops
    - link capacity & propagation delay vary
  - dynamic sources: flows join and leave
  - arbitrary patterns: aperiodic or random packet arrivals. Only constraint is the TSpec {burst, rate}.

  → Similar to the Internet

- Overall framework
  - Avoid time-synchronization
  - Decouple the latency guarantee problem from the jitter guarantee problem
  - Latency guarantee
    - Regulators
  - Jitter guarantee
    - Latency guaranteed network & Time-stamping & Buffering

Problem statement

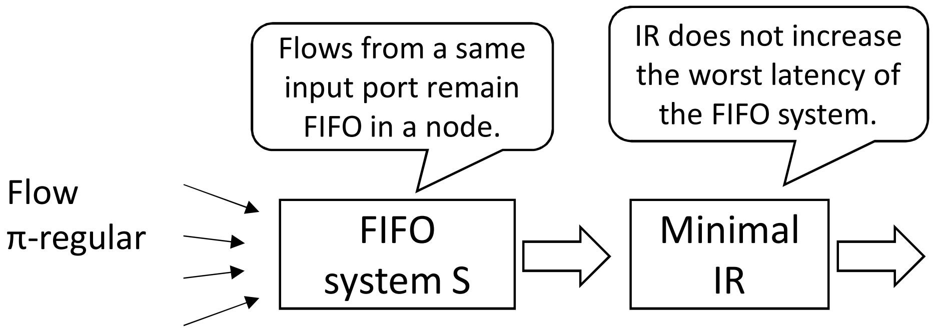
- The Internet already has the DiffServ framework for latency guarantee.
  - Provided that in every link the total high priority traffic rate does not exceed the link capacity
    - Resource reservation & Admission control mandatory
  - It works well for lightly utilized networks.
- However, when utilization is medium to high; the burst accumulates.
  - Assume *n* identical flows coming into a switching node, each with {B, r}
  - At every node the burst accumulates as much as
    - If a shared queue with a FIFO scheduler → $Bout \le B+(n-1)B*r/C$. (Note that nr/C is the utilization.)
    - If queue per flow with a packet-based fair scheduler → $Bout \le B+(n-1)L*r/C$. (L is the max packet length.)
  - The flows are now with {Bout, r}.
  - This accumulation continues as flows travel.
  - A cycle in network topology acts as a feed-forward loop. → "Burst explosion"

Solution candidates

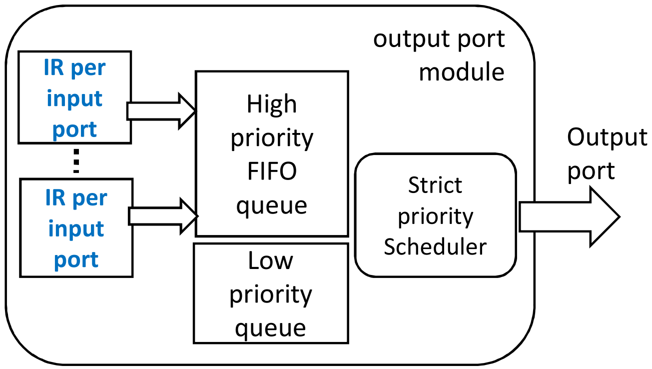
- Solutions to mitigate burst accumulation:
  - Slotted operation (without strict synchronization)
  - Packet metadata based forwarding (e.g. Latency budget, etc.)
  - Flow regulation: Forcing a flow into its initial shape {B, r}
- Their shortcomings
  - The slotted operation or the cyclic queuing, strict or loose, can be seen as an example of regulation with {B, r, and the start phase}. However,
    - it requires the slot planning and the source cooperation,
    - the cycle-time can be as large as the accumulated burst size, because it may have to accommodate all the other flows in its path.
  - Metadata-based forwarding
    - may not disperse the accumulated burst.
    - requires lookup/decide/queue-reorder/overwrite in line speed.
  - Regulation
    - requires flow state maintaining. → We argue this can be overcome with flow aggregation.

Latency guarantee framework

- Regulation on Flow aggregate
  - **ATS**
    - At every node
    - IR per input port
    - IR has only one queue, but still requires individual flow states
  - FAIR
  - PFAR
  - Other possible solutions



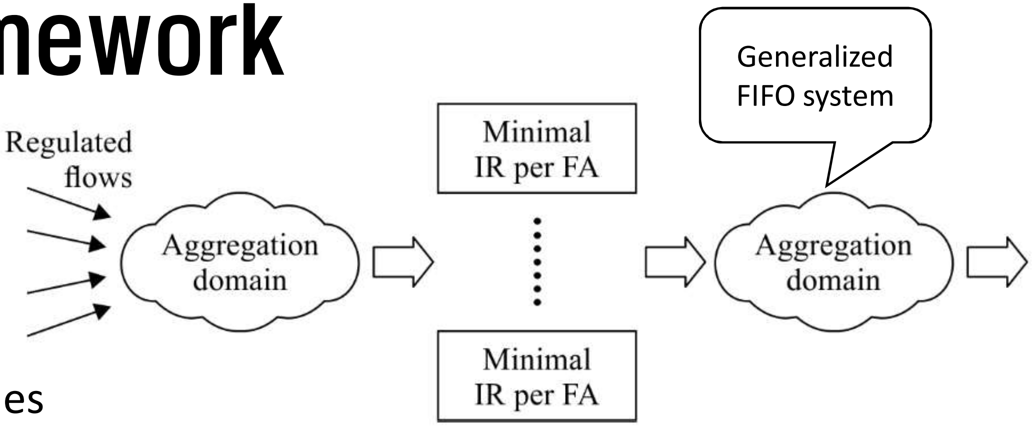
IR architecture



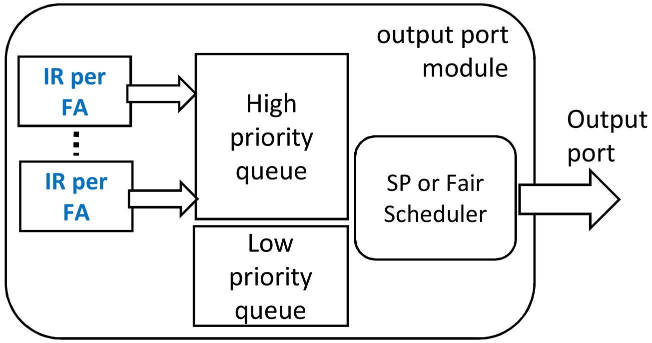
Implementation practice of ATS

Latency guarantee framework

- Regulation on Flow aggregate
  - ATS
  - **FAIR** (Flow aggregate & IR)
    - At “aggregation domain (AD)” boundaries
    - FA is of flows with same path in AD
    - IR per FA
    - Generalized ATS
    - Shown to work better than ATS [FAIR]
  - PFAR
  - Other possible solutions



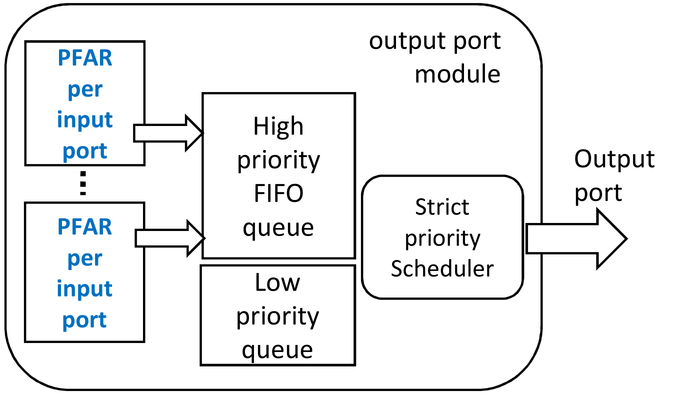
Generalized IR architecture



Implementation practice of FAIR
at an AD ingress

Latency guarantee framework

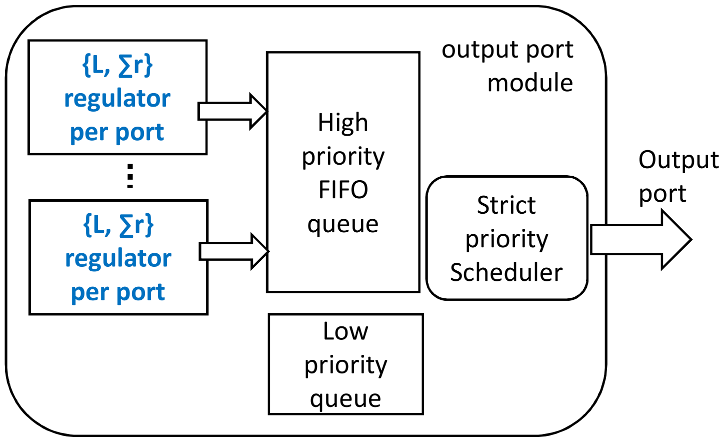
- Regulation on Flow aggregate
  - ATS
  - FAIR
  - **PFAR** (Port-based FA regulation)
    - At every node or at critical links to break the cycle
    - FA is of flows having same input/output port of a node
    - Regulate FA, not individual flow, with {∑B, ∑r}
    - Best scalability: no need to maintain individual flow states
    - Shown to work almost as well as ATS [ADN].
  - Other possible solutions



Implementation practice of PFAR

Latency guarantee framework

- Regulation on Flow aggregate
  - ATS
  - FAIR
  - PFAR
  - **Other possible solutions**
    - More strict regulation than the TSpec (e.g. regulate with {L, ∑r} per input port, at every node ≈ Slotted operation)
    - Using the forwarding metadata (e.g. unsynched source timestamp) in the packets to reproduce the initial inter-arrival process (e.g. at every node)

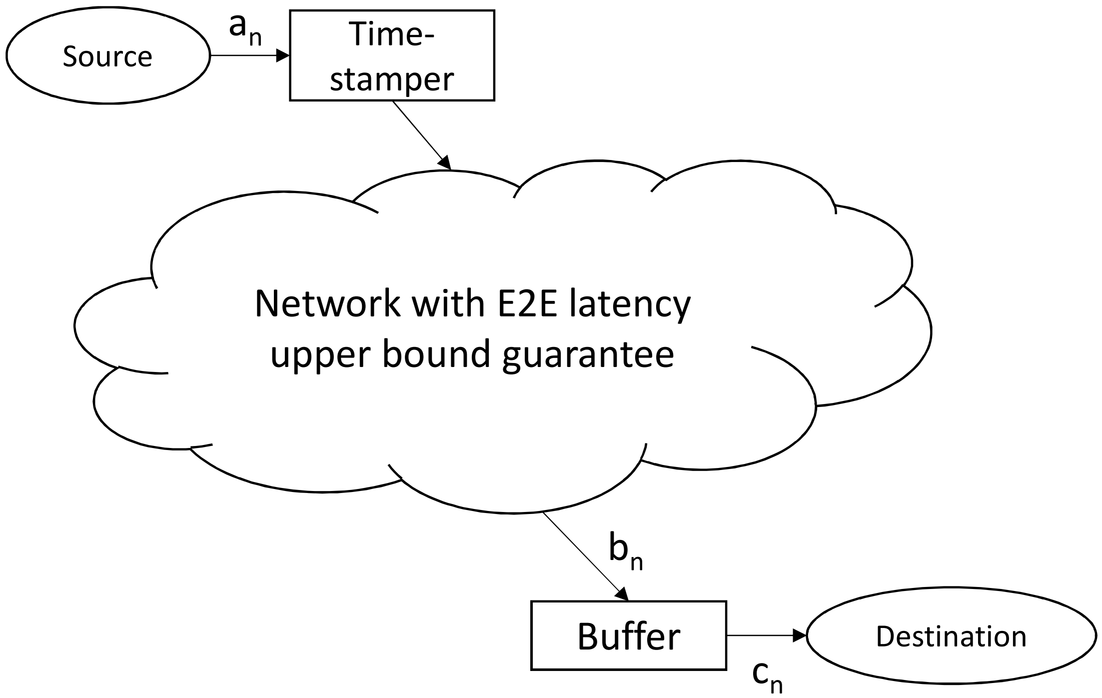


Implementation practice of {L, ∑r} regulator

Jitter guarantee framework

- Jitter guarantee ≈ Reproducing the inter-arrival process with the inter-departure process of a network.

- With a latency guaranteed network, time-stamping and buffering at the network boundary:
    - E2E jitter is upper bounded.
        - It can be set to zero.
    - 'E2E buffered latency' ($c_i - a_i$) is also upper bounded.
    - Moreover, we can control the jitter bound. We can even have zero jitter, with E2E buffered latency bound ≈ 2* E2E latency bound [BN].



a_n : the arrival time of n_{th} packet of a flow

The jitter between packets i and j is defined as $|(c_i - a_i) - (c_j - a_j)|$.

Thank you

- Please take a look at

  https://datatracker.ietf.org/doc/draft-joung-detnet-asynch-detnet-framework/

- Comments and Questions are welcome!