Asynchronous Deterministic Networking (ADN) Framework for Large scale networks

draft-joung-detnet-asynch-detnet-framework-01

Red letters: added contents

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

IETF 115

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
  - Decouple the latency guarantee problem from the jitter guarantee problem
  - Latency guarantee
    - Regulators or Metadata based forwarding
  - Jitter guarantee
    - Latency guaranteed network & Time-stamping & Buffering

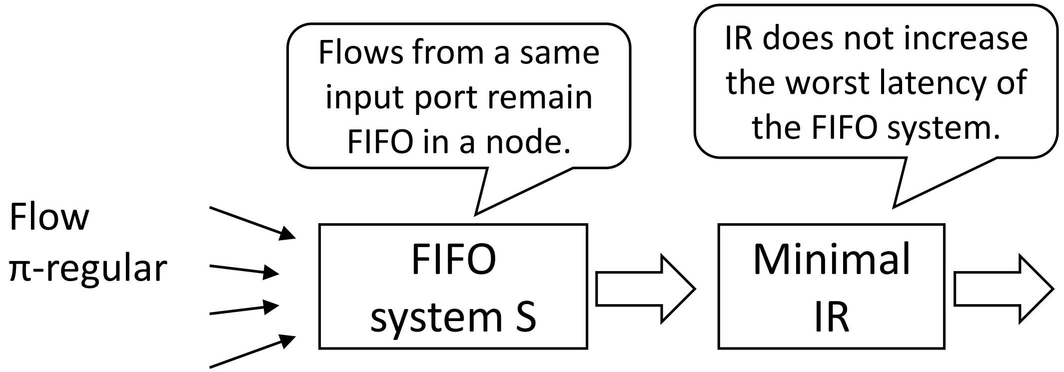
Solution candidates & shortcomings

1. Flow regulation: Forcing a flow into its initial shape {B, r}
   - requires flow state maintaining. → This can be overcome with flow aggregation.
2. Packet metadata based forwarding
   - may require lookup/decide/queue-reorder/overwrite in line speed.
     → This can be compensated by the performance advantage of stateless fair-queuing at core nodes.
3. Slotted operation (without strict synchronization)
   - can be seen as an example of regulation with {Burst, rate, and start phase},
   - 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.

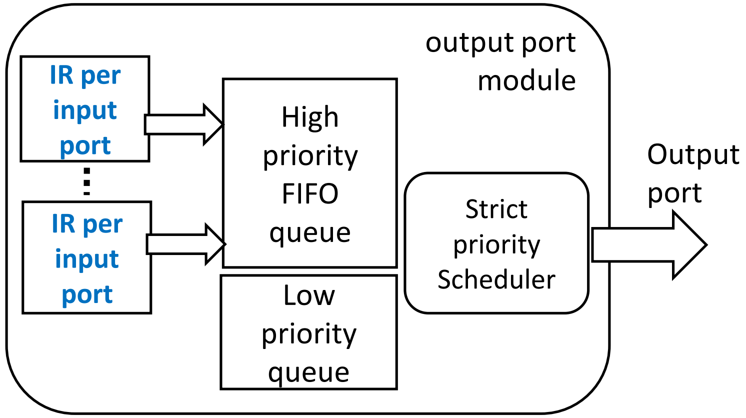
- The proposed solutions in this document include 1 and 2.

Latency guarantee framework with regulators

- 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



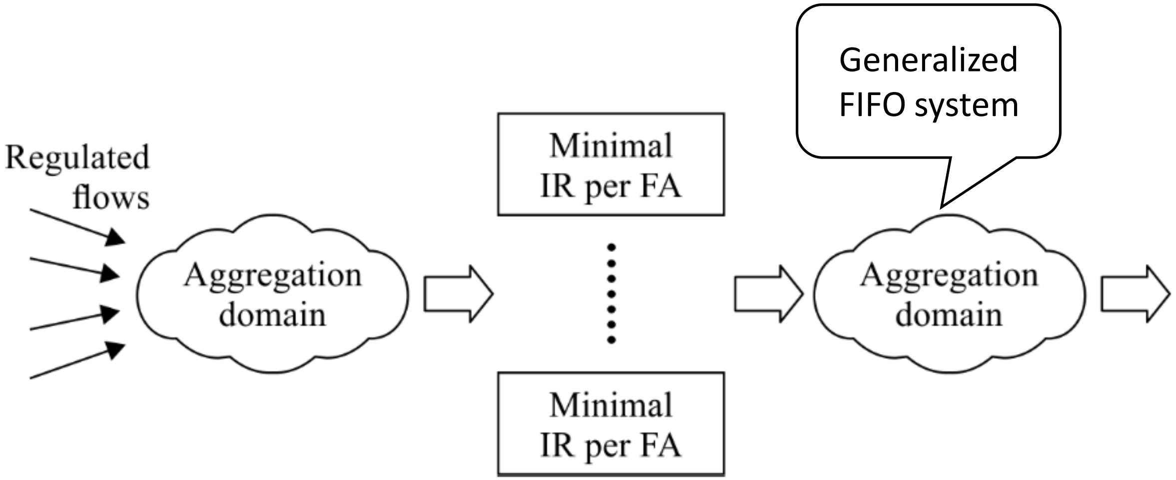
IR architecture



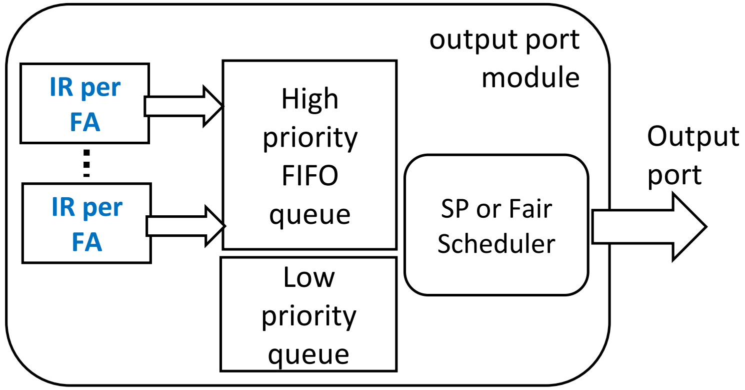
Implementation practice of ATS

Latency guarantee framework with regulators

- 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



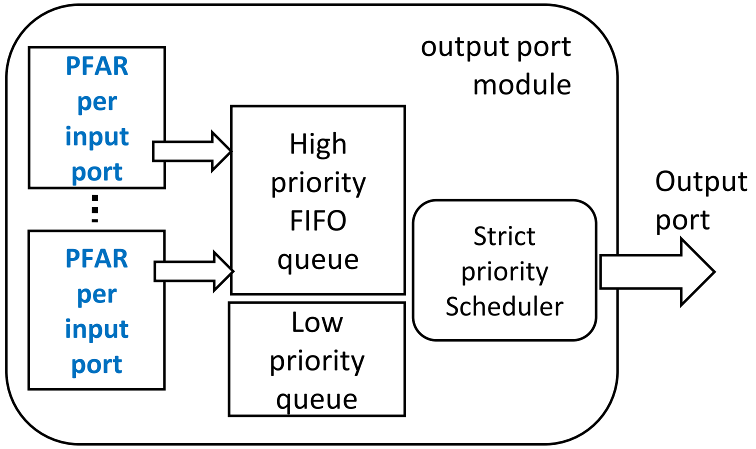
Generalized IR architecture



Implementation practice of FAIR
at an AD ingress

Latency guarantee framework with regulators

- 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 with metadata

BACKGROUND

- Fair queuing (e.g. Virtual Clock [Zhang])
  - is based on **FT**, Finish time F(p) = Service finish time of packet p in an Ideal fluid model = Service order in a realistic packet-based model. Smaller FT gets earlier service.
  - FT is determined by the “fair distance” from the previous packet’s F(p-1) in the same flow, or from the packet's arrival time:

    $F(p) = \max\{F(p-1), A(p)\} + L(p)/r;$
  - It requires F(p-1) to get F(p). F(p-1) is the “flow state”.
- We propose to use fair queuing in core nodes without flow state. Necessary conditions are:
  - Within a flow,
    1. Keep the fair distance between FTs of consecutive packets
    2. Preserve the actual service completion order
    3. Reflect the time lapse as hops progress: $F_h(p) \geq F_{h-1}(p)$
  - Across the flows,
    4. Align the FTs to the current time

Symbol	Definition
Node	An output port module of a switching device
$F_h(p)$	‘Finish time’ of packet p at node h
$A_h(p)$	Arrival time of packet p at node h
L(p)	Length of packet p
r	Flow service rate

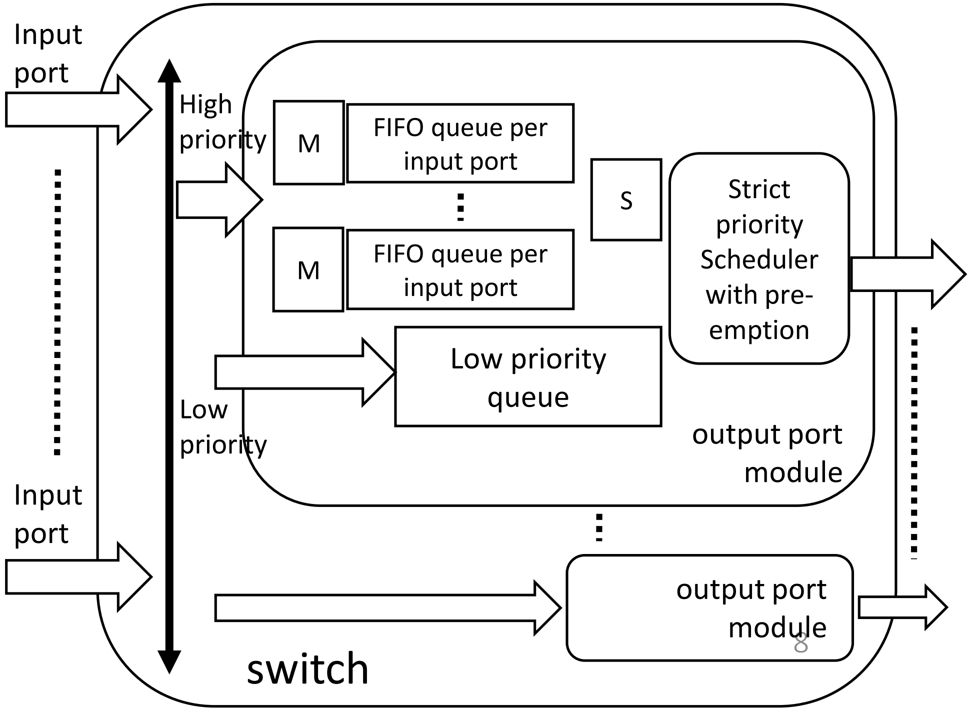
Latency guarantee framework with metadata

Symbol	Definition
Node	An output port module of a switching device
$F_h(p)$	'Finish time' of packet p at node h
$A_h(p)$	Arrival time of packet p at node h
$L(p)$	Length of p
r	Flow service rate
$d_h(p)$	**FT increment factor of p at node h**

Solution: Global FT based forwarding framework

1) Obtain $F_0(p)$ at the entrance node 0, as in the Virtual Clock:
   $F_0(p) = \max\{F_0(p-1), A_0(p)\}+L(p)/r.$
2) in a core node, increment FT of previous node by $d_h(p)$:
   $F_h(p) = F_{h-1}(p) + d_{h-1}(p).$
3) $d_h(p)$ is a non-decreasing function of p within a node busy period & should be larger than or equal to the actual delay;
   $d_h(p) \geq A_{h+1}(p) - A_h(p).$
4) In a core node, preserve the service order of packets from the same input port.

- By 1) ~ 3), the conditions 1 ~ 4 are met.
- By 4), using the per-input port FIFO queue is possible.
- The metadata to carry in a packet: $F_h(p)$, $d_h(p)$.
  - These are dynamic and need to be updated.
  - $d_h(p)$ can be set to d_h. Then metadata update is simpler.



M: Finish time (F) marker

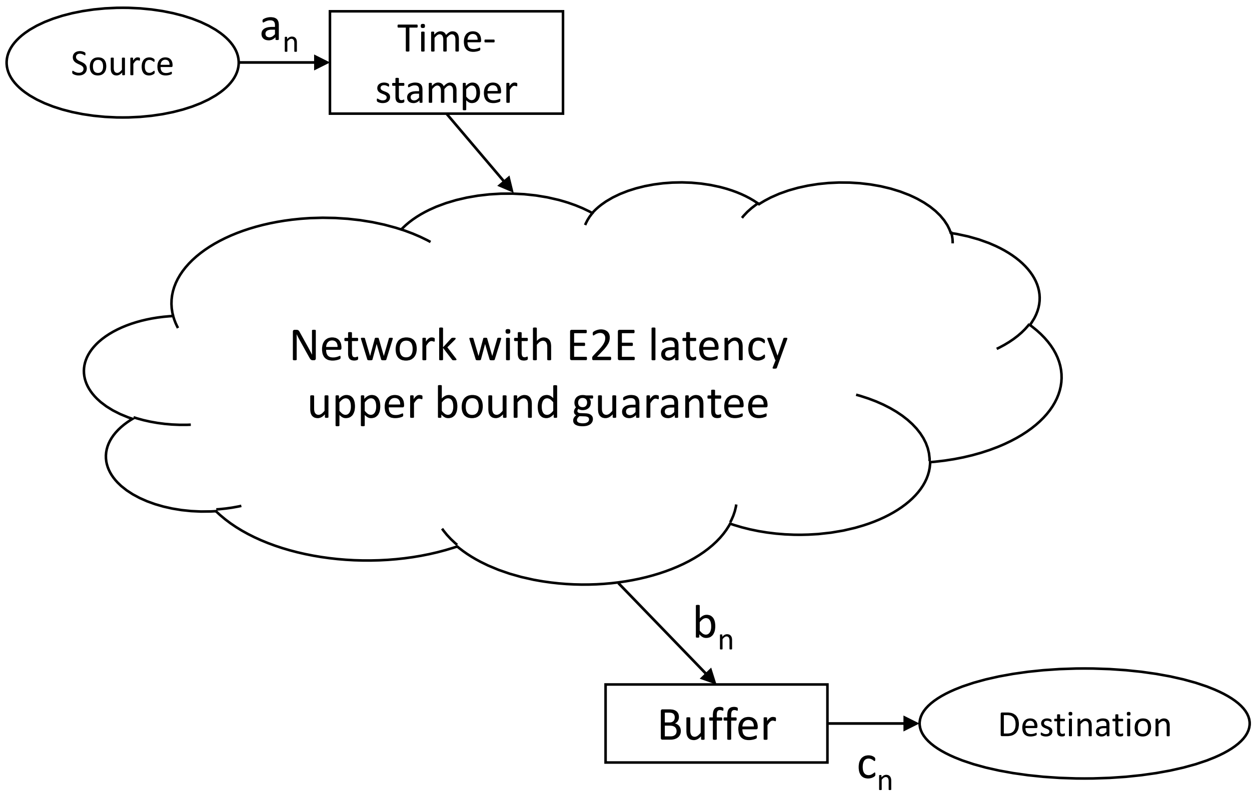
S: HoQ examine, select the min F, update $d_h(p)$

Discussion

- Simple FIFO implementation & simple metadata management.
- d_h can be obtained (theoretical or measured) in a distributed manner; or by a central network manager then distributed.
  - As an example d_h can be u_h, the maximum latency in node h for any flow.
- A packet with max latency up until h gets $F_h(p) = F_0(p)+(A_h(p)-A_0(p))$, while others have $F_h(p') > F_0(p')+(A_h(p')-A_0(p'))$; therefore does not delayed more than it would in a stateful VC.
- The proposed solution is work conserving, contrary to the non-work conserving scheme [Stoica].
- It approximates packetized rate proportional servers (PRPS) [Stiliadis] whose E2E delay bound is bounded with $\leq B/r + H*(L/r+L_{max}/C)$,
  - where B is the max burst of the flow, H the number of hops, C the link capacity, L the max packet length of the flow, L_{max} the max packet length of all the flows.
  - Note that the bound is free from other flows' bursts. Flow protection can be achieved.

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!

- [FAIR] Jinoo Joung. "Framework for delay guarantee in multi-domain networks based on interleaved regulators." Electronics 9, no. 3 (2020).
- [ADN] Jinoo Joung, Juhyeok Kwon, Jeong-Dong Ryoo, and Taesik Cheung. "Asynchronous Deterministic Network Based on the DiffServ Architecture." IEEE Access 10 (2022).
- [Zhang] Lixia Zhang. "Virtual clock: A new traffic control algorithm for packet switching networks." In *Proceedings of the ACM symposium on Communications architectures & protocols*, pp. 19-29. 1990.
- [Stoica] Ion Stoica and Hui Zhang. "Providing guaranteed services without per flow management." *ACM SIGCOMM Computer Communication Review* 29, no. 4 (1999): 81-94.
- [Stiliadis] Dimitrios Stiliadis and Varma Anujan. "Rate-proportional servers: A design methodology for fair queueing algorithms." IEEE/ACM Transactions on networking 6, no. 2 (1998): 164-174.
- [BN] Jinoo Joung and Juhyeok Kwon. "Zero jitter for deterministic networks without time-synchronization." IEEE Access 9 (2021).