SIEMENS

Ingress Policing for TSN Streams

IEEE 802 Plenary Meeting – July 2014, San Diego

Feng Chen, Siemens AG
Franz-Josef Goetz, Siemens AG

SIEMENS

Contents

1. Recap: Why Ingress Policing for TSN
2. A Description of Typical Error Patterns
3. Ingress Policing Strategy
4. Case Study with Simulations: using per Stream Ingress Policing on Edge Ports to Protect TSN against Babbling Talkers

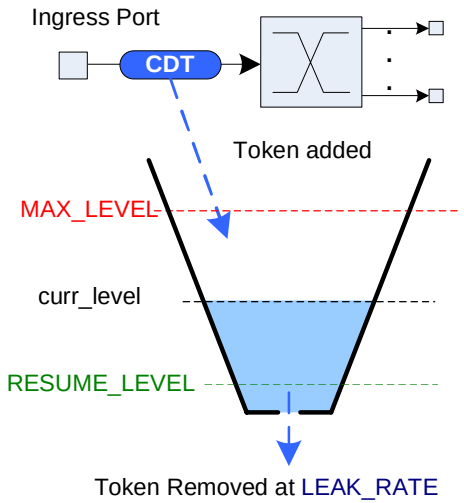
SIEMENS

Recap: Why is Ingress Policing Necessary for TSN

- Ingress policing detects and eliminates traffic overload at ingress ports at an early stage by dropping frames of involved Traffic-Classes or Traffic-Streams
  - avoid exhaustion in buffer resources
  - prevention of potential traffic overload at egress ports
  - guarantee low latency and provide robustness for CD-streams
- Ingress policing is needed especially for **preventing error propagation** in TSNs, which can be caused by
  - a babbling or misconfigured talker producing higher traffic load as reserved
  - a babbling bridge transmitting the same streams multiple times
  - …
- Note! Only streams reserved by MSRP will be handled by ingress policing, while unreserved streams due to misconfiguration or other reasons must be dropped by TSN bridges.
  - e.g. a misbehaved TSN bridge forwarding streams over a wrong communication path due to choosing a wrong destination port or misconfiguration

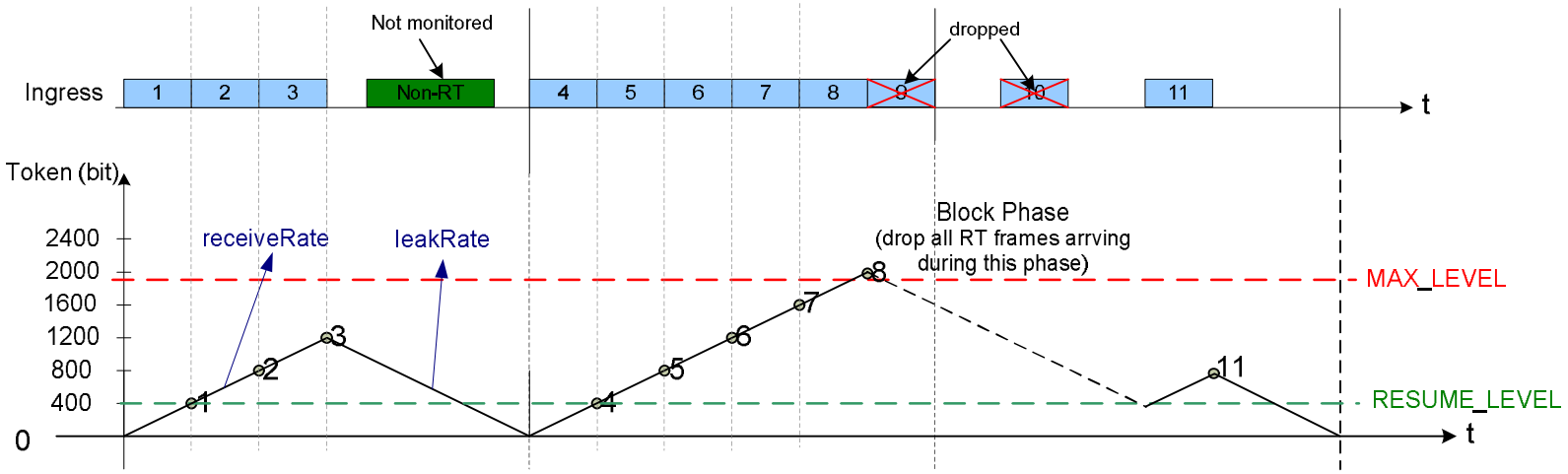
http://www.ieee802.org/1/files/public/docs2013/bv-goetz-TSN-GuaranteedLatency4CDT-20130904-v1.pdf

Proposal: Ingress Policing w/ Leaky Bucket



Ingress policing with Leaky Bucket

- Perform traffic policing on target streams at the ingress ports
- Tokens are increased with a receiveRate and decreased with a leakRate
  - both rates are calculated based on bandwidth reserved for target streams
- Enter **block phase**, when curr_level > MAX_LEVEL at the end of reception
  - frames arriving during the **block phase** will be **dropped***
- Deblock when curr_level < RESUME_LEVEL
- Monitor the length of each received frame and drop those whose lengths exceed a specified **MAX_FRAME_SIZE**



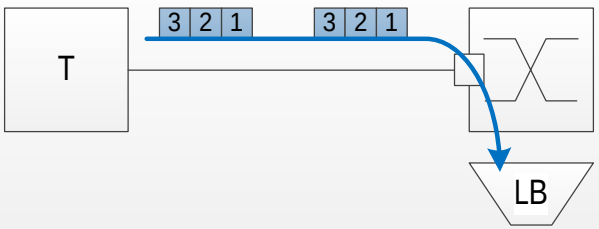
*** Temporary blocking instead of permanent blocking is applied to provide E2E connectivity**

Ingress Policing Strategy
Question 1: per Stream, per Group or per Class?

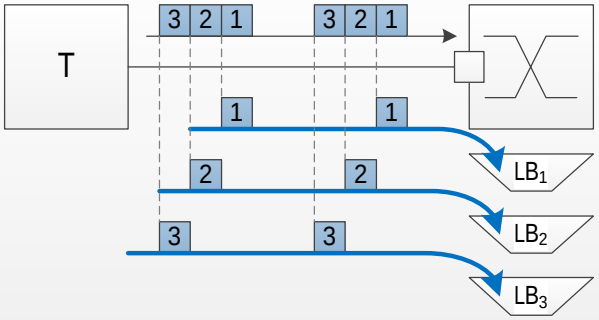
- **Stream** is identified by SR-DA and Priority
  - e.g. one controller creates a sendlist: (C->D_n, C->D_{n-1} ... C->D_2, C->D_1), where there are a total of n streams (S_n, S_{n-1} … S_2, S_1)

- Three options for ingress policing
  - per class: one leaky bucket for all streams (of the same CDT class)
  - per stream: one leaky bucket per stream
  - per group: a selected set of streams share the same leaky bucket

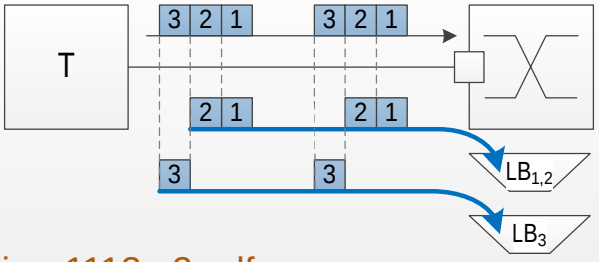
Ingress policing per port



Ingress policing per stream



Ingress policing per group

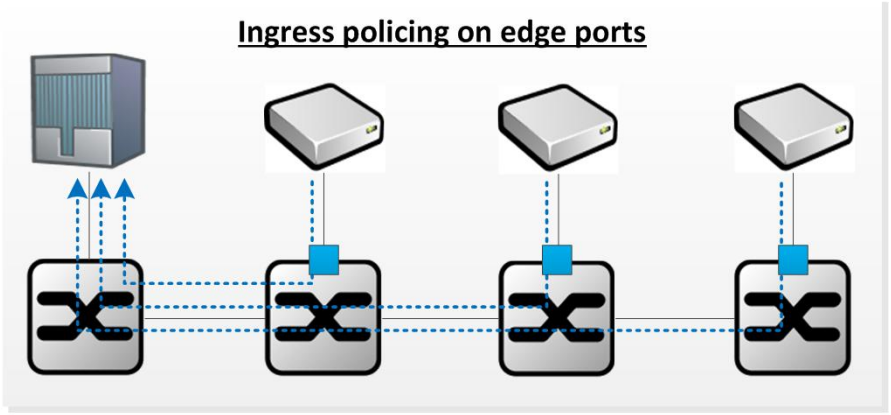


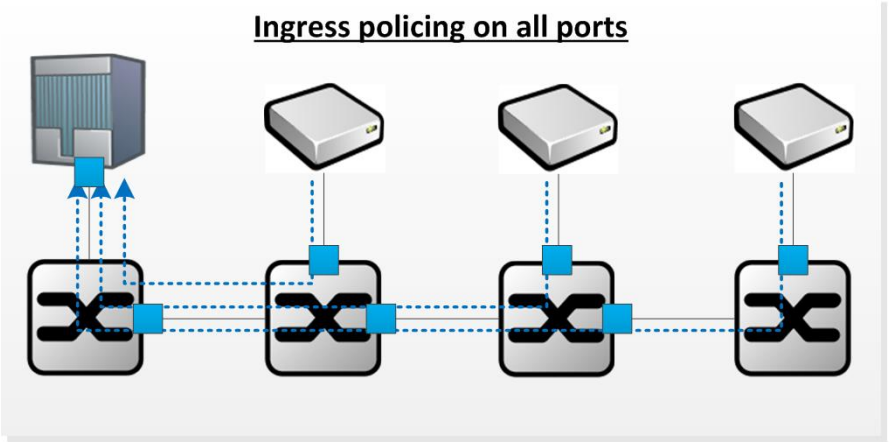
http://www.ieee802.org/1/files/public/docs2013/tsn-jochim-ingress-policing-1113-v2.pdf

Ingress Policing Strategy
Question 2: on Edge Ports or on All Ports?

- **Edge port** is defined relative to a given stream to be a bridge port with a direct link to the talker that generates that stream

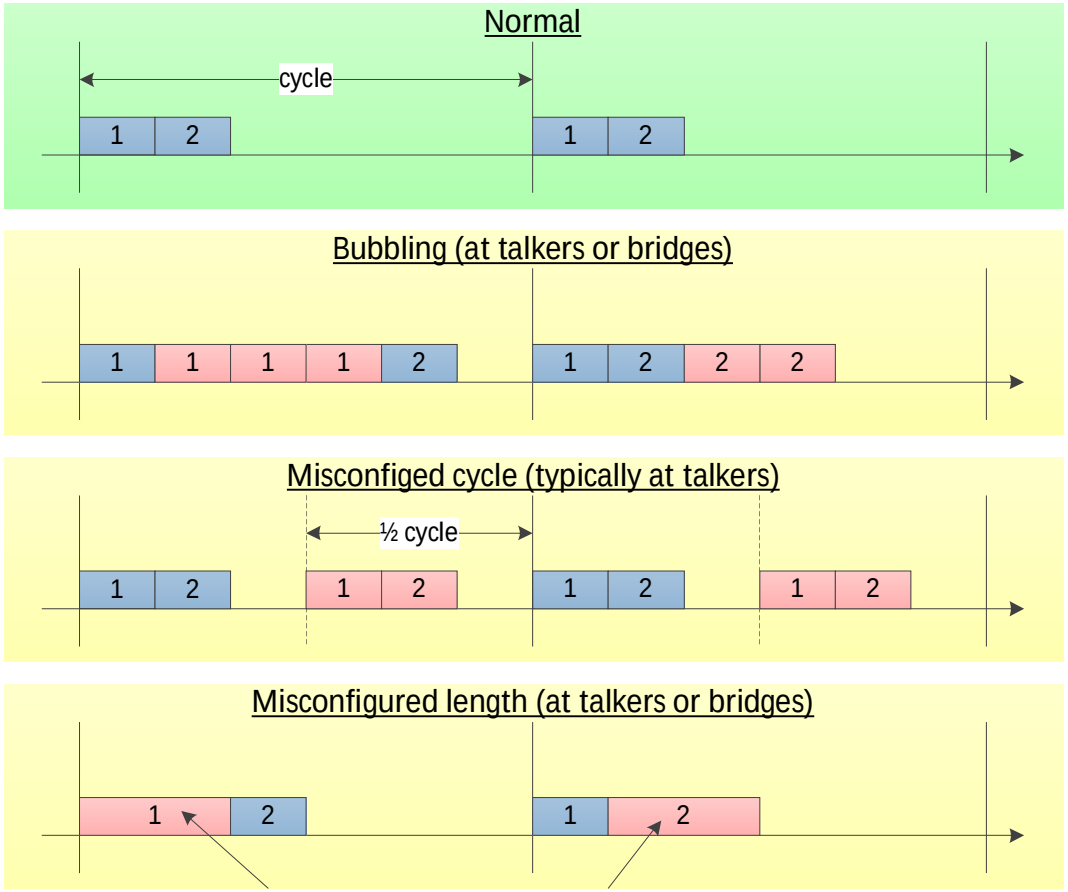
- Two options
  - only on edge ports
  - on all ports receiving target streams





Possible Error Patterns

- At talkers
  - babbling talkers: generating the same frame multiple times
  - misconfigured cycle: generating streams with more or less frequent periods
  - misconfigured payload length
- At bridges
  - babbling bridges: forwarding the same frame multiple times
  - misconfigured adding bytes (tags, padding) leading to wrong frame length
- ***any other typical error patterns??***



Frames with abnormally long length can be filtered off by specifying MAX_FRAME_LENGTH in leaky bucket

SIEMENS

Case Study 1: Misbehaved Talkers

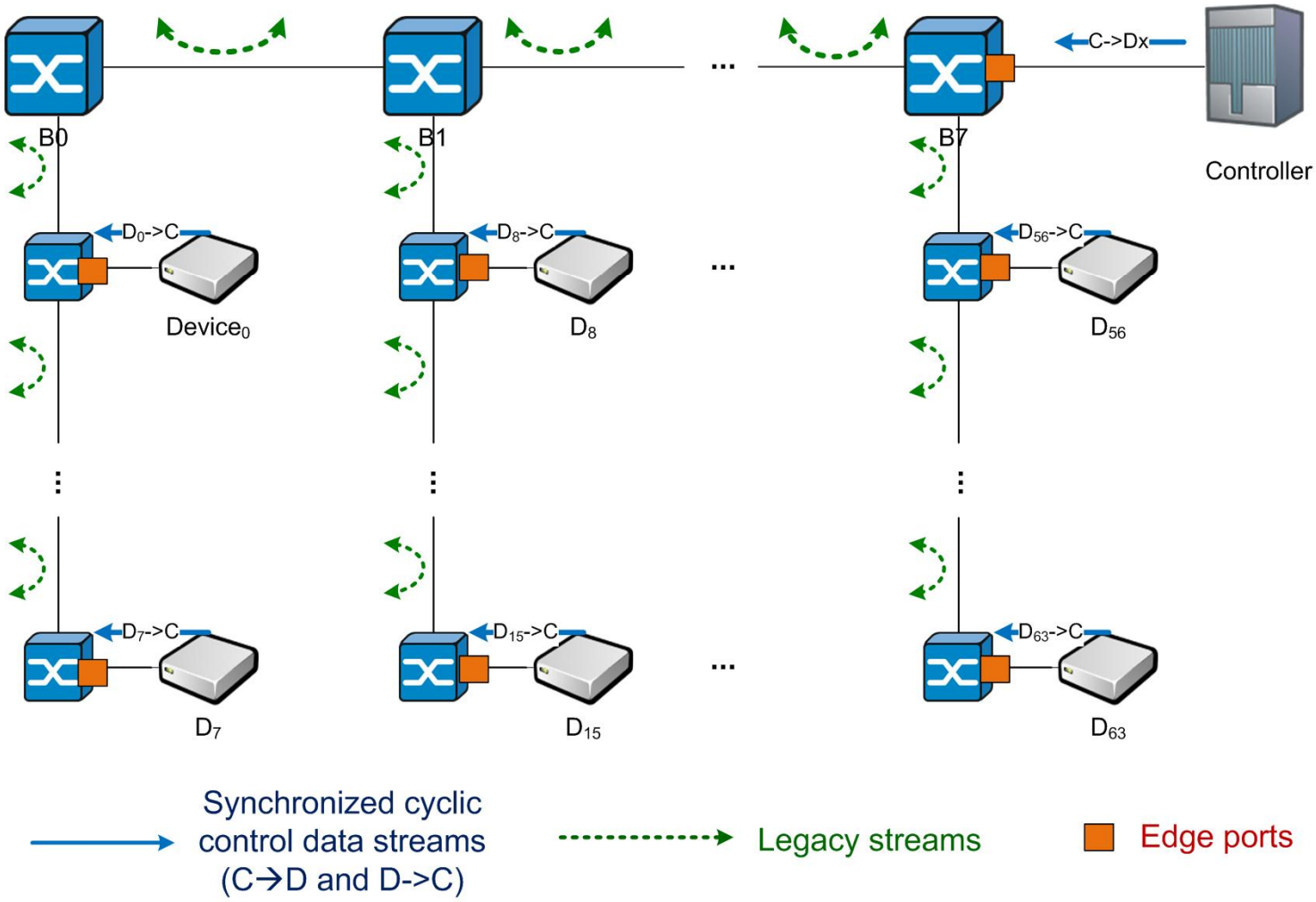
- Different error scenarios require different ingress policing strategies

- This case study focuses on misbehaved talkers, which randomly generate babbling streams

- Investigate how **per stream ingress policing at edge ports** can
  - guarantee latency
  - minimize error propagation
  - help reduce impact of faulty streams on other non-faulty streams

- We conduct simulations with our TSN bridge model in OMNEST

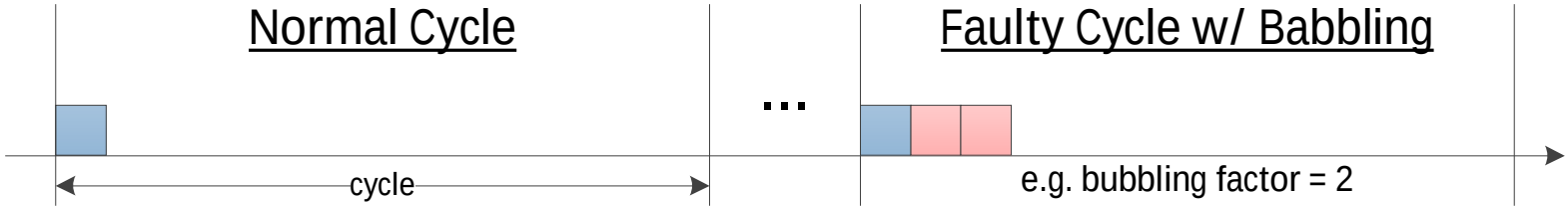
Test Bench:
One Controller (C) <-> 64 Devices (D) in a Comb Topology



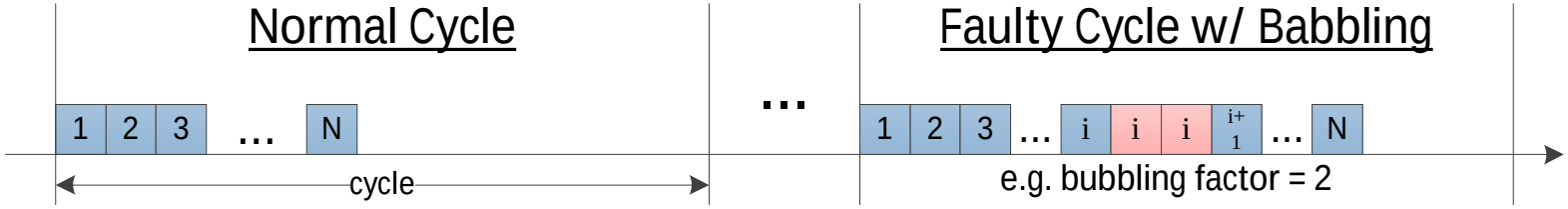
SIEMENS

Configuration of Error Model (Babbling Talker)

Error pattern at devices for each D->C stream



Error pattern at controller for C->D streams

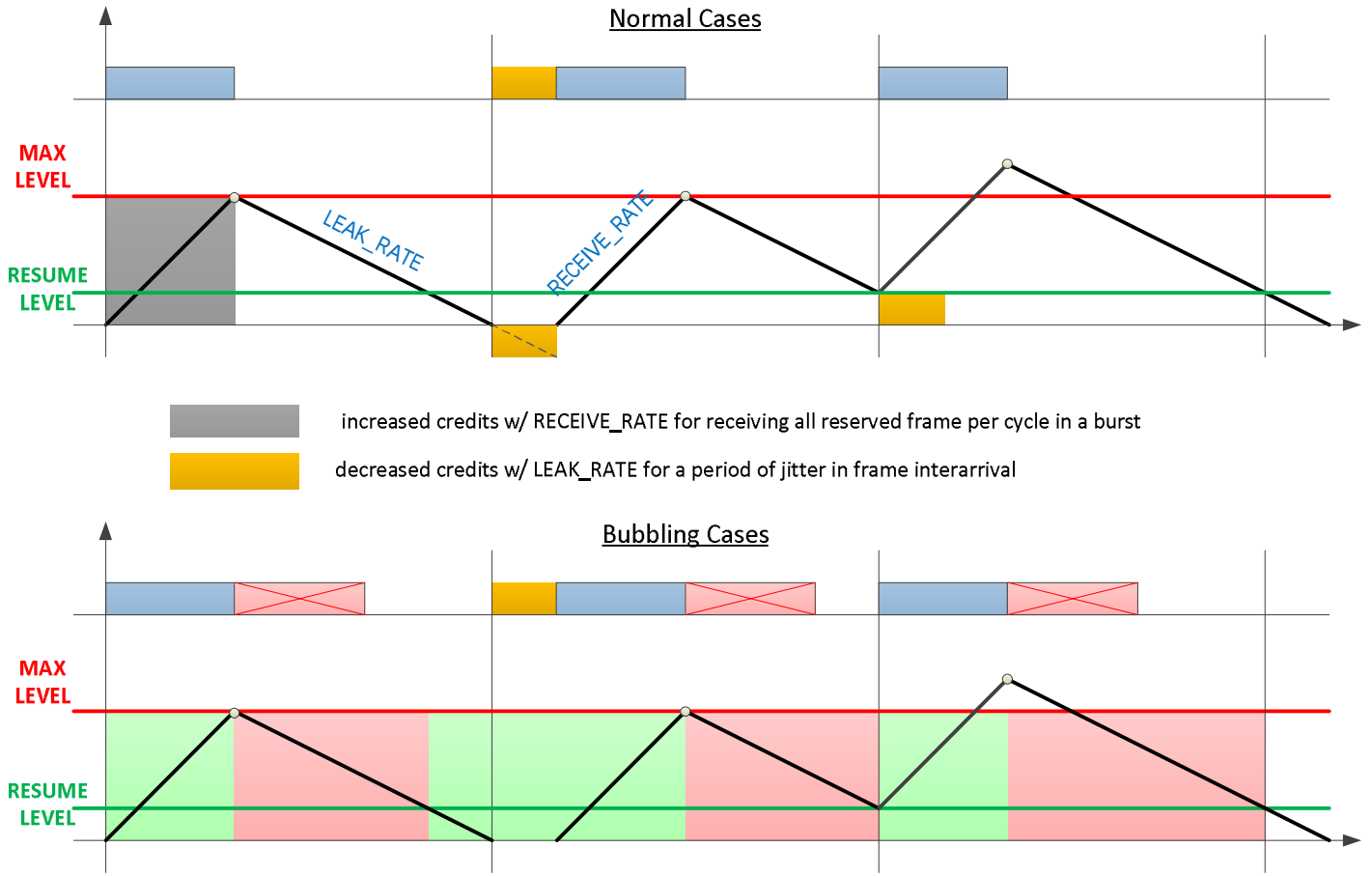


General Settings

Parameter Settings	
Bit-rate	1 Gbps
IFG/PRE+SFD	12 Bytes/ 8 Bytes
Control App Cycle Time	500 µs
Control Data Traffic	each CD stream has a constant control data frame size, which is initially chosen from a distribution: (10% 64 Bytes, 10% 512 Bytes, 80% between 128~384 Bytes); sending order for C->D streams is optimized using specific engineering tool; all streams are generated and put in the transmission queue at the beginning of each cycle
Legacy Traffic	each legacy stream has frames of different sizes following a distribution (25% 1536 bytes, 25% 64 bytes, 50% mean = 750B) with 30% traffic load
Link Delay	fixed for each link, randomly chosen between 50 ns ~ 500 ns
PHY Tx/Rx Delay	125 ns/ 125 ns
Bridge Delay	fixed for each bridge, randomly chosen between 400 ns ~ 850 ns
Transmission Mode of CDT	cut-through
Preemption	Enabled, CDT is preemptive, legacy traffic is pre-emptable
Error Prob. of Babbling Talker	in the range 1% ~ 10%
Babbling Factor	1: double, 2: triple (in misbehaved cycle randomly occurring according to error prob., a stream with a doubled or tripled number of frames will be generated)
Ingress Policing Strategy	**per stream, only on edge ports of CDT talkers**
Shaper	guaranteed highest priority for CD-Traffic, without egress shaping

Configuration of Ingress Policing Filter per Stream on Edge Ports

- In this case study, each CDT stream contains only one frame per cycle



Simulation Results

For 64 Devices -> Controller Streams

Bubbling Factor	Error Prob.	w/o Ingress Policing	with Ingress Policing at Edge Ports	
		Max. Makespan (µs)	Max. Makespan (µs)	Drop Rate*
1	1%	160.6	151.5	100%
	2,50%	168.1	151.5	100%
	5%	179.7	151.5	100%
	7,50%	184.6	151.5	100%
	10%	190.2	151.5	100%
2	1%	177.5	151.5	100%
	2,50%	185.6	151.5	100%
	5%	208.6	151.5	100%
	7,50%	218.6	151.5	100%
	10%	229.9	151.5	100%
0	No Errors	151.5	151.5	0%

For Controller -> 64 Devices Streams

Bubbling Factor	Error Prob.	w/o Ingress Policing	with Ingress Policing at Edge Ports	
		Max. Makespan (µs)	Max. Makespan (µs)	Drop Rate*
1	**	155.5	155.5	100%
2		159.4	159.4	100%
0	No Errors	151.5	151.5	0%

* Drop Rate is calculated as: number of dropped frames / number of more generated frames by babbling talkers

** Only one randomly chosen stream is babbling in every faulty cycle

SIEMENS

Summary

- For networks with babbling talker errors, ingress policing per stream on edge ports can provide satisfying results, which are described as follows
  - all more generated frames can be dropped without being propagated into the network
  - latency of CDT (measured in makespan) will not be affected by the babbling streams in case that each talker has only one stream.
  - if one talker contains more than one streams, latency of CDT (measured in makespan) may be increased (because one bubbling stream may delays other non-babbling streams sent behind it already at the talker). However, applying such ingress policing schemes can protect streams of other talkers (measured using in another makespan) from being affected. (The second case will be further investigated in a scenario with more than one control applications)

- Further simulations will be conducted to investigate error cases including bubbling bridges
  - Ingress policing only at edge ports may not be sufficient
  - More complex schemes for configuring leaky buckets are needed

Thank you for your attention!

Feng Chen

I IA ATS TM5 1

Gleiwitzer Str. 555

90475 Nürnberg

Phone: +49 (911) 895-4955

Fax: +49 (911) 895-3762

E-Mail: chen.feng@siemens.com

siemens.com/answers