



# *Developing High Availability Features to Address Failure Recovery of the RNIF*

April 2004

# Authors

- **Suresh DAMODARAN, RosettaNet/Sterling Commerce**
- **Takashi TOKAE, RosettaNet Japan/Fujitsu**
- **Kenji NAGAHASHI, RosettaNet Japan/Fujitsu**
- **Shigenori NOMURA, RosettaNet Japan/Sony EMCS**
- **Hiroaki ISAKA, RosettaNet Japan/NTT Communications**
- **Kazuto KIMURA, RosettaNet Japan/Intel**

# Contents

- Business Case
- Communication Failures and responses
- Server Overload related Failures
- Why Retry Algorithm wouldn't do
- Pacing Algorithm
- Permanent Failure and Response

# Business Case

- RNIF implemented in production in 100s of sites
- Some of them require high volume (upwards of 50K messages per hour)
- Consistently supporting high volume (rate) requires highly available RNIF implementations
- In high volume, RNIF implementations get communication failures
- Problem not unique to RNIF implementations – SOAP based messaging specifications (EbXML Messaging, WS-I BP), and pure MIME based (EDIINT AS2)
- This problem is a symptom of RNIF success 😊

# Communication Failures

## *The many ways to fail*

### COMMUNICATION FAILURES

While communicating RNIF messages with trading partners (TP) failure in communication can occur between two RNIF implementations

- Transfer Protocol failure (e.g., HTTP)
- (RNIF) message failure (e.g., absence of an expected Receipt Acknowledgement)
- (RNIF) implementation failure (e.g., a server hosting the RNIF implementation fails)

# Communication Failures

## *Attributes*

### **FAILURES ARE RELATIVE to a TP**

- While TP A may perceive a failure at TP B, TP C may not perceive a failure at TP B in the same time period!

### **ANTICIPATED or UNANTICIPATED**

- Anticipated system shutdowns or system reconfigurations at TP A can create (unanticipated) failures when TP B communicates to TP A

### **BEFORE or DURING**

- Failures may exist prior to creating a PIP<sup>®</sup> (Partner Interface Process<sup>®</sup>) based business transaction (PIP Instance), or may occur during the execution of a PIP Instance

### **ONE or BOTH**

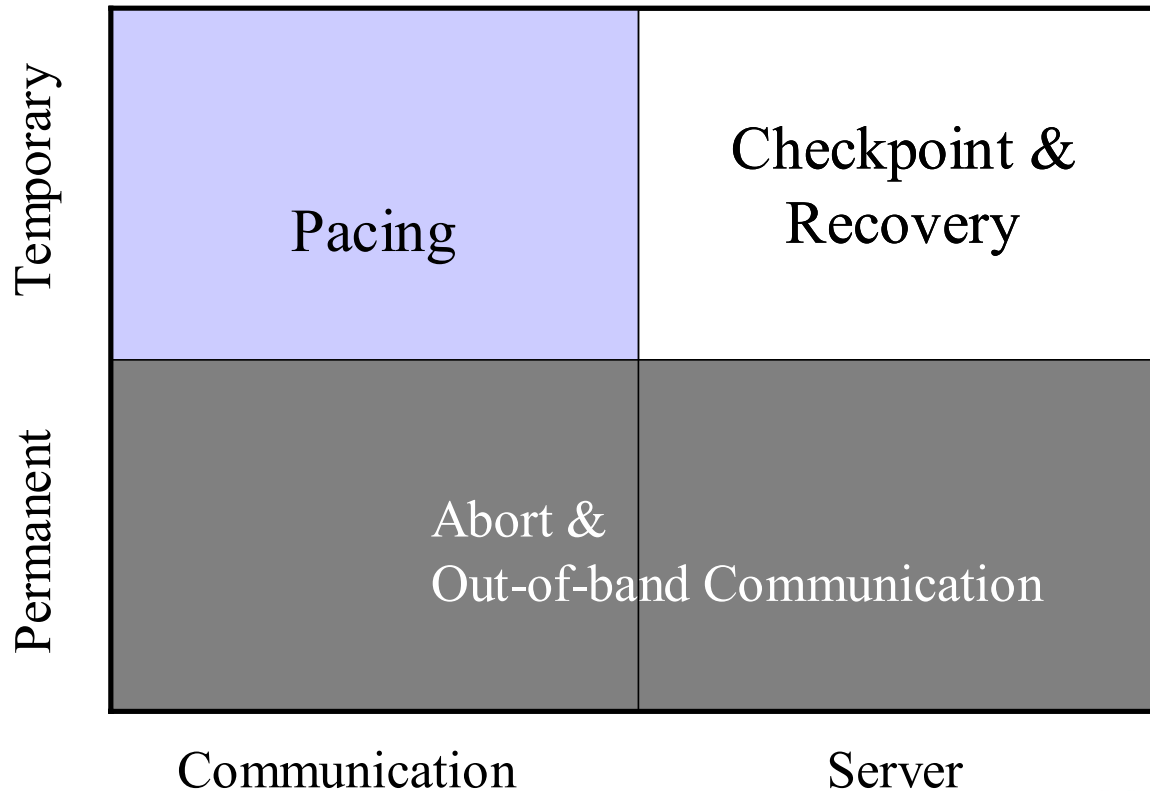
- One or both of sender and receiver may fail during a PIP message exchange

# Communication Failures

## *Detecting Permanent Failures*

### PERMANENT FAILURE

Recovery is possible only by using “out-of-band” communication.  
We focus on Temporary Communication failures today.



# Transfer Protocol Failure

## *Only HTTP Considered*

### **HTTP Response Codes 200 and 202**

- In RNIF2.0 [2], two HTTP response codes, 200 or 202 (Section 2.4.2.2 "Processing Inbound HTTP Posts") MUST be returned for a successful HTTP communication. Response code 200 is used for synchronous HTTP requests and 202 is used for asynchronous HTTP requests.

### **HTTP Response Codes 3XX, 4XX, and 5XX**

- RNIF 2.0 states that "3xx, 4xx and 5xx error conditions must be dealt with in the usual way, governed by the local policy."

### **Focus on HTTP Responses 502, and 503**

- This presentation focuses on how to deal with in a standard way when 502(Service Temporarily Overloaded), and 503 (Service Unavailable) are received.

# Current Potential Behavior

## *HTTP Response Code 502, 503*

### Message Receiver

- Free to discard the received message (implications of sending 502/503!)

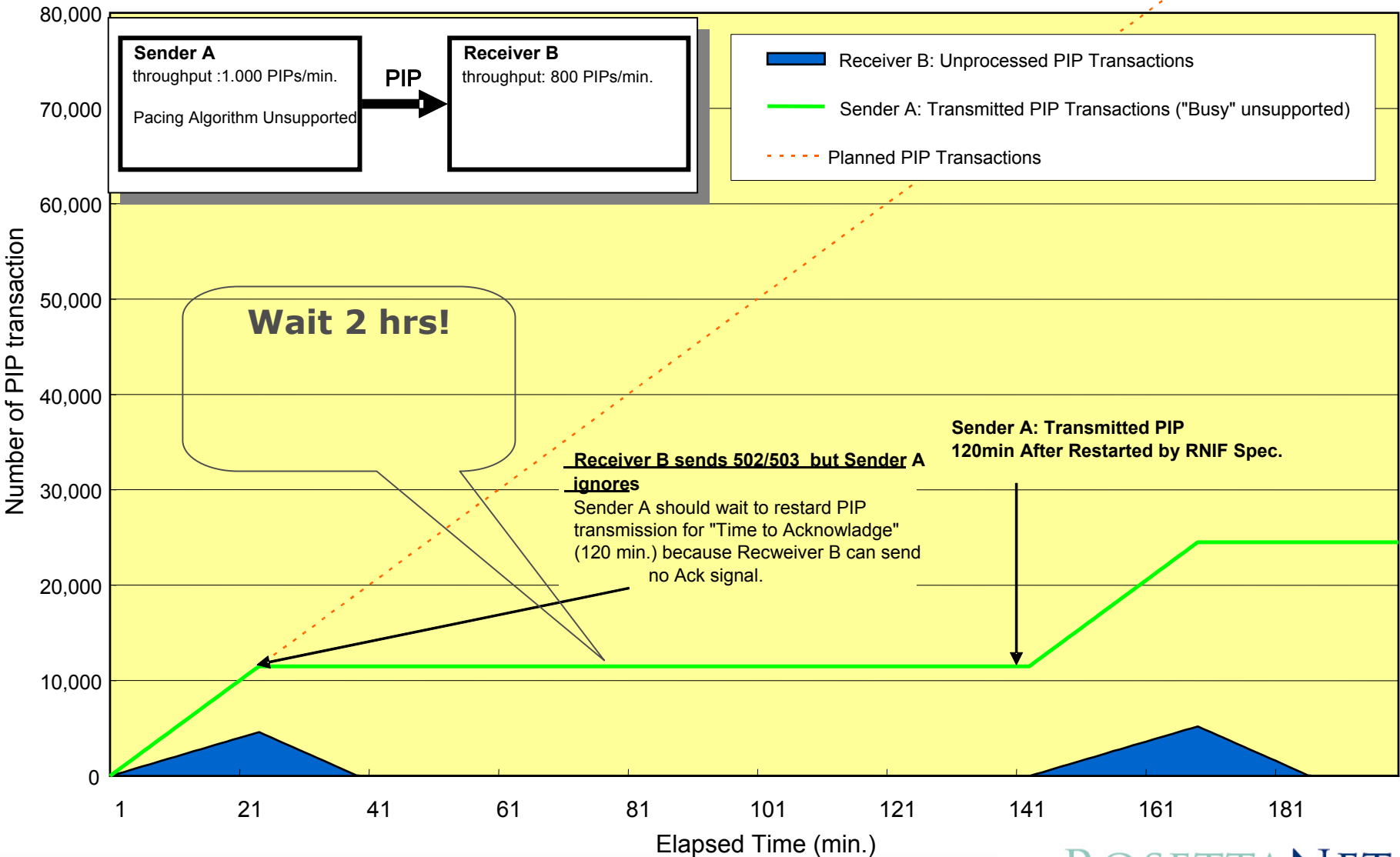
### Message Sender

- Ignore 502/503 from Receiver. Pretend nothing happened **(WRONG!)**

OR

- Assume a permanent communication failure has occurred, and send a Notification of Failure (PIP 0A1), and a human reinitiates the same PIP Instance **(Untrue assumption, many times!)**, OR
- Assume a temporary communication failure has occurred, and initiate a retry using retry algorithm.

# Using Retry Algorithm



# Retry Algorithm is not Enough

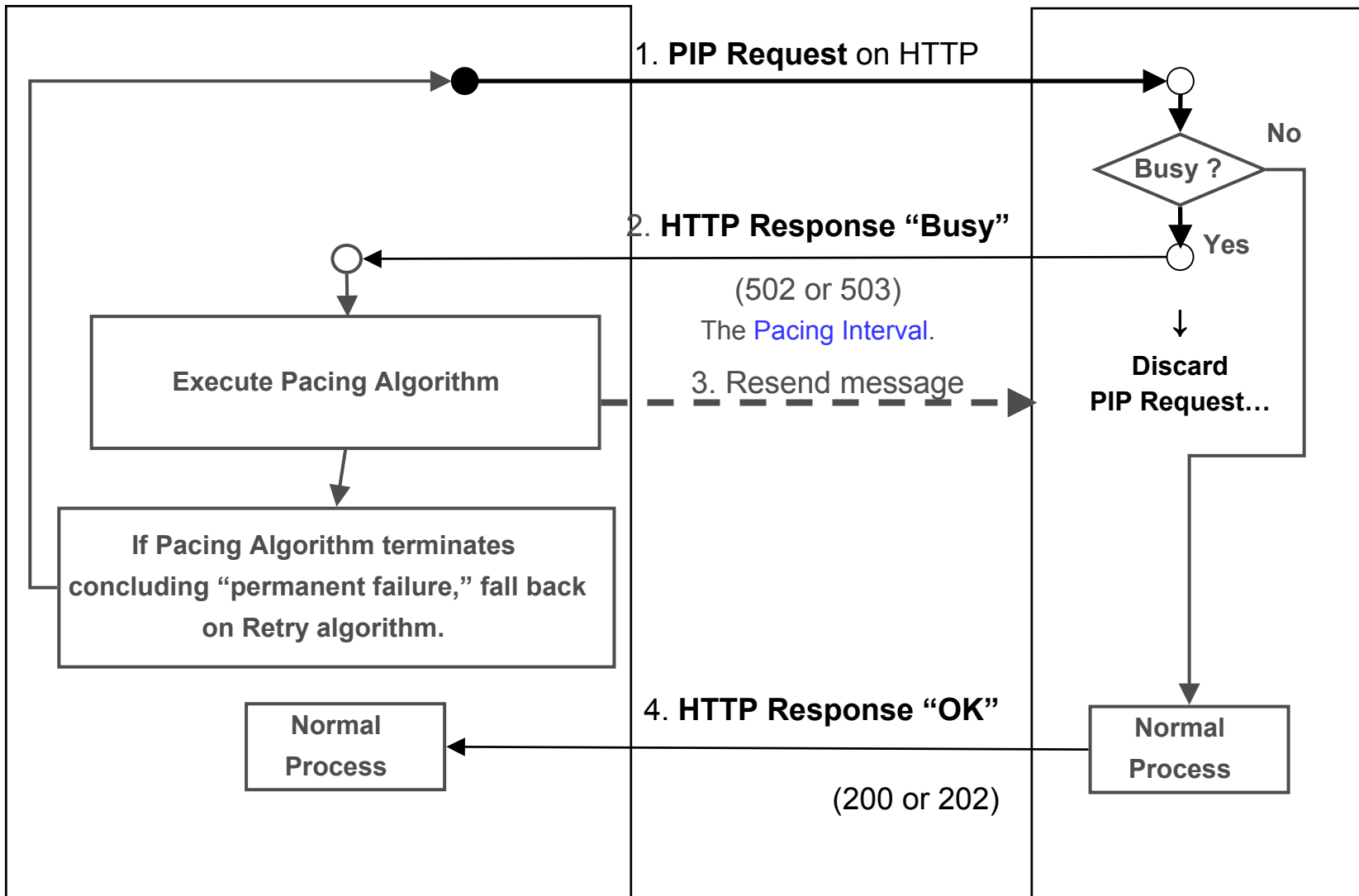
- Retry Algorithm not intended for HTTP failures, rather for non-receipt of Receipt Acknowledgement or RNIF Exceptions
- When the HTTP Server is busy, or unavailable, no Receipt Acknowledgement can be sent!

**So, we need a “Pacing Algorithm” mainly to work between consecutive retries**

# Retry Algorithm vs. Pacing Algorithm

Sender A

Receiver B



# Retry Algorithm vs. Pacing Algorithm

Retry Algorithm	Pacing Algorithm
For non-receipt of Receipt Acknowledgement or RNIF Exceptions	For server overload
Only for Action message	For Action and Signal messages
Retry Interval = Time-to-Acknowledge = 2hrs	Pacing Interval = 5 min (typically)
Retry Count = 3	Pacing Count = 10 (typically)

$\text{Pacing Interval} * (\text{Pacing Count} + 1) < \text{Time-to-Acknowledge}$

# Pacing Algorithm

## Context

### PACING ALGORITHM

To recover from HTTP failures of 502 or 503 kinds, Pacing Algorithm is executed after an Action or Signal message is sent.

### PACING INTERVAL & PACING COUNT

**Pacing Interval:** Interval between two consecutive resend of messages during Pacing Algorithm execution

**Pacing Count:** Maximum number of times a message is resent during Pacing Algorithm execution

# Pacing Algorithm

## *Execution*

### Preconditions

1. TP A has defined Pacing Interval and Pace Count
2. TP A receives HTTP error code 502 or 503 from TP B.
3. TP A is the Sender and TP B is the Receiver

# Pacing Algorithm

## Execution

### STEPS

1. TP A resends the message (action or signal) "Pace Count" times, every such resend happening after the elapse of a "Pacing Interval". Thus, the last message resend will happen at (Pace Interval \* Pace Count) period after the initial receipt of the error code at TP A.
2. If TP A continues to receive either HTTP error code 502 or 503 for all resends during this instance of the Pacing Algorithm, or receives no responses to the resends, TP A concludes that TP B has a permanent failure. On the other hand, may TP B responds with a normal reaction (a Receipt Acknowledge, Exception, or a Response message). In either case, the Pacing Algorithm is concluded.
3. If a Notification of Failure (PIP 0A1) arrives from TP B during step 1, then the Pacing Algorithm is concluded.
4. While this algorithm is executing, no new PIP Instances must be initiated by TP A to send to TP B. However, pending action messages (only Response Action messages) and signals may be sent to TP B.

# Pacing Algorithm

## Execution

### Post Condition

TP A concludes TP B has a communication failure, and whether it is permanent or temporary, OR

TP A concludes TP B has no communication failure (with a signal or PIP 0A1)

### Notes

It is possible that both TP A and TP B are executing a Pacing Algorithm at the same time. Since a Pacing Algorithm is guaranteed to conclude, the end result can be either a successful conclusion of the PIP Instance, or a failure at one or both TP from the other TP's perspective.

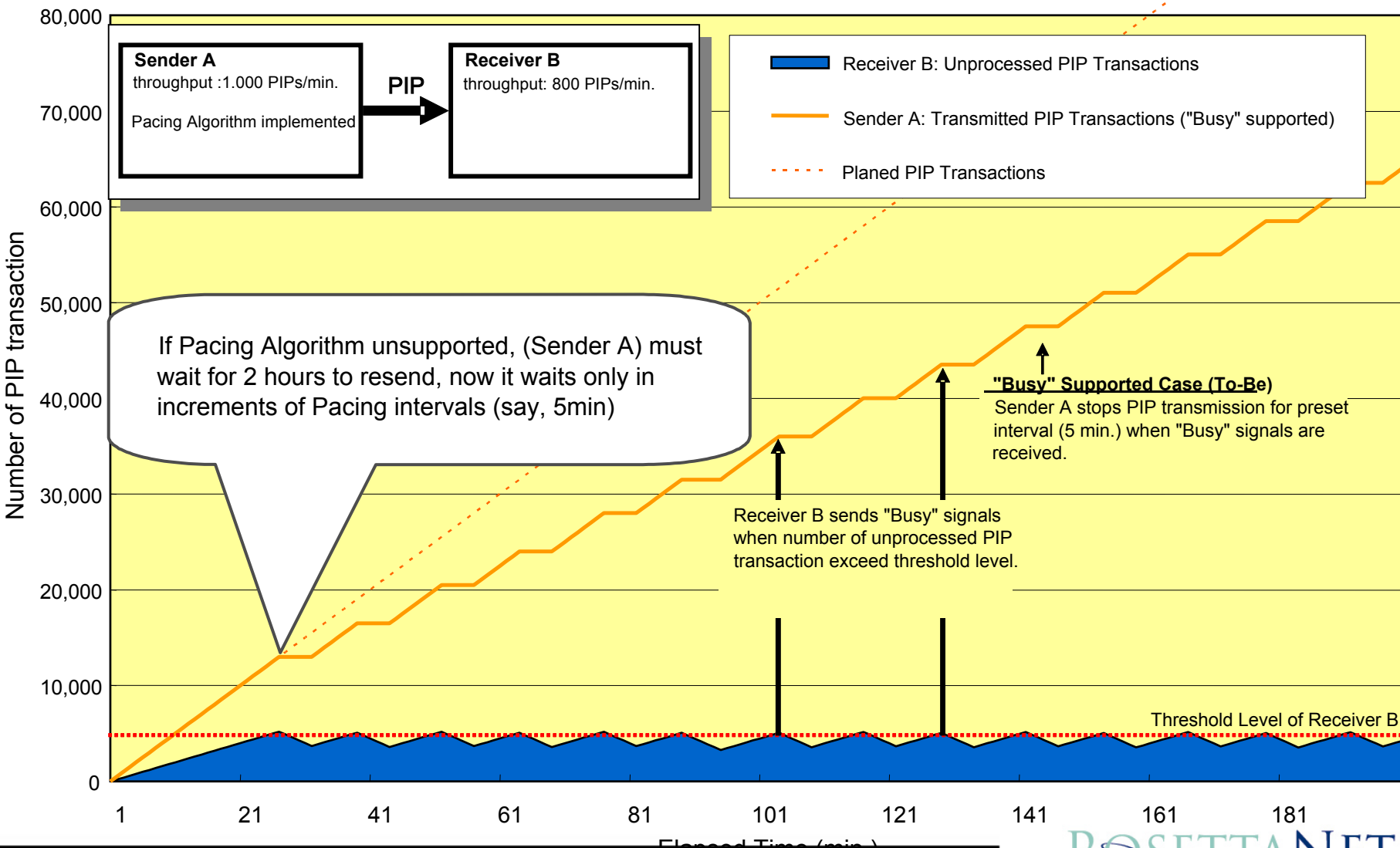
If Pacing Algorithm executes during sending PIP 0A1, then do not send another PIP 0A1 to the same TP right after!

If the Pacing Algorithm concludes a permanent communication failure, retry based on Retry message is to be started 2hrs after the send prior to invoking Pacing Algorithm

Pacing Interval and Pacing Count must be agreed between partners

Pacing Algorithm is only for Asynchronous (HTTP) Messages

# Pacing Algorithm Implemented by Sender



# Permanent Failure

## *Reaction*

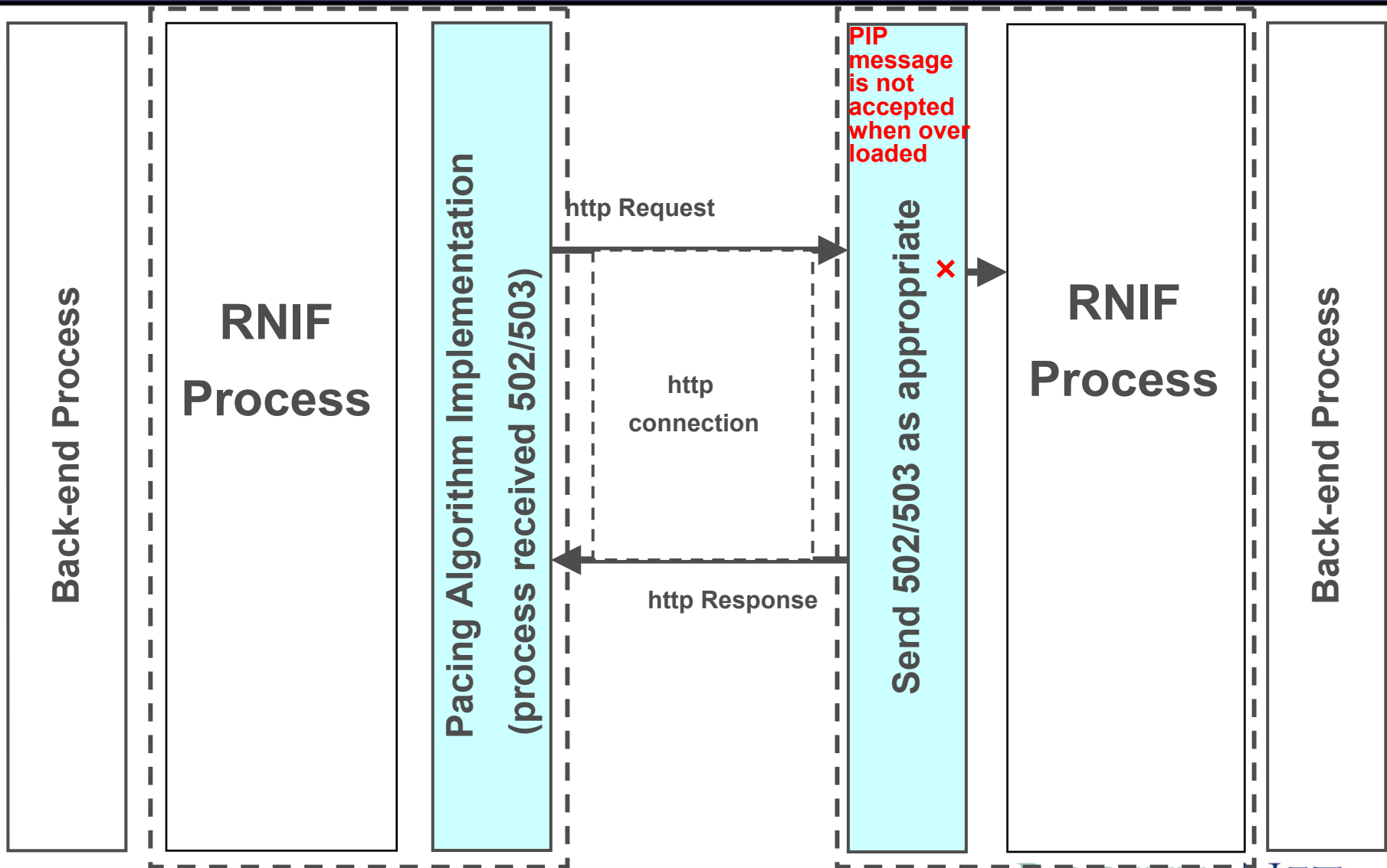
### **Out-of-band communication for Notification of Failure**

- To “reset/abort” the PIP Instance unilaterally
- PIP 0A1, Phone/fax/...
- The communication (PIP 0A1, phone, fax,...) must be processed by the receiver to be successful – just receiving is not enough!

# Implementation of Pacing Algorithm

Sender A

Receiver B



# Comparing Protocols

## *Retry and Pacing*

**RNIF 2.0**

**Yes**

**Yes**

**eBMS 2.0**

**Yes**

**No**

**WS-I BP**

**?**

**No**

**AS2**

**Basic**

**No**

**Retry**

**Pacing**

# Future Improvements

- Variable Pacing Interval
  - Exponential back off
- Pacing Interval communicated in HTTP Response Header/Signal

*Thank you!*

*Questions?*