intdash®

White Paper

intdash の機能概要

30th May, 2018

aptpod

intdash の機能概要

intdashの製品群では、前章にて明らかにした課題を解決するために、以下のような機能を実装しています。本章では、intdashが実装する機能の概要について説明します。

intdashの機能

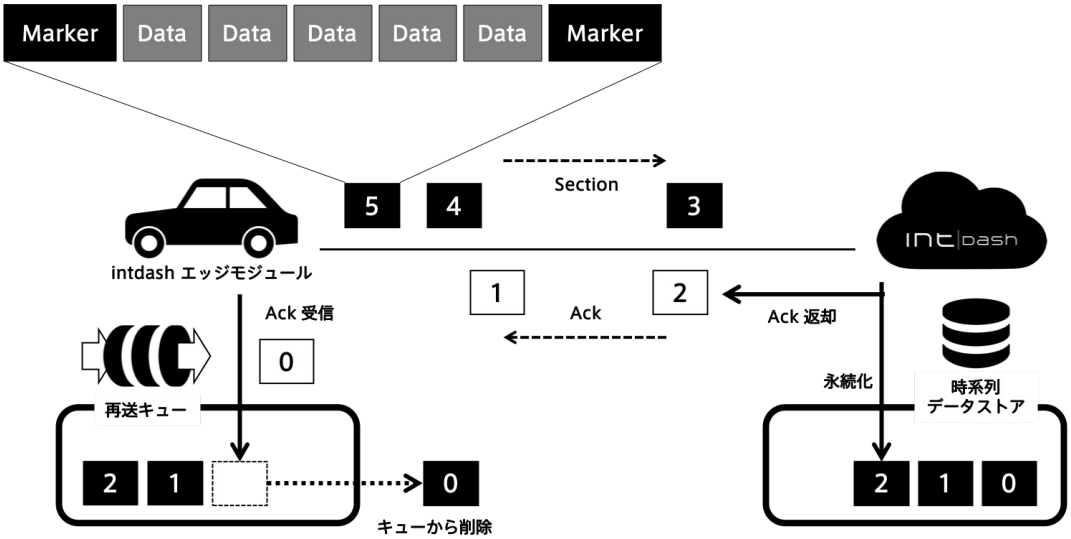
- **発生データの完全回収**
    - 再送シーケンスにより、回線切断時にもデータを欠損なく永続化します
- **効率伝送**
    - 伝送データを効率よくパッキングし、狭い帯域でも大量のデータを送信します
- **送信データの流量抑制**
    - 変動する帯域に合わせて伝送データの流量を調整し、優先データを確実に伝送します
- **経過時間による時間管理**
    - デバイスのクロック同期状態に依らない、データ発生時刻の管理方法を提供します

発生データの完全回収

intdashは、デバイスで発生した高頻度データを欠損なく永続化するための機能を備えています。

intdashにおける発生データの完全回収は、エッジモジュールがもつ再送キューと永続化が完了した際にサーバから返送されるACKメッセージにより実現されます。エッジデバイス側で発生したデータは、サーバへの送信と同時に再送キューに保存され、サーバからACKメッセージが返却されるまではエッジデバイス側で保持されます。このキューの存在により、物理回線やコネクションが切断した場合でも、データは欠損しません。また、トランスポート層のプロトコルと異なり、ACKメッセージはサーバでデータの永続化が完了した後に返却されるため、データがサーバに到達したことだけでなく、永続化されていることが保証されます。

intdashにおける欠損管理は、発生したデータごとではなく、セクションとよばれるデータセット単位で実行されます。発生したデータは随時サーバへ送信されますが、ある程度長い（1秒〜10秒程度）間隔ごとにセクションマーカーとよばれるセクションの開始／終了を表す目印を挿入することにより、セクションごとに欠損の有無を判定します。このように、ある程度長期のデータセットごとに欠損管理を実施することにより、TCPにおける遅延ACKと同様の効果を生み出し、リアルタイム性を犠牲にすることなく発生データの完全回収を実現します。



エッジモジュールによって再送処理が管理されることで、エッジモジュールを利用するクライアントプログラムは、回線状況を気にする必要がなくなり、アプリケーション固有のロジックに集中できます。

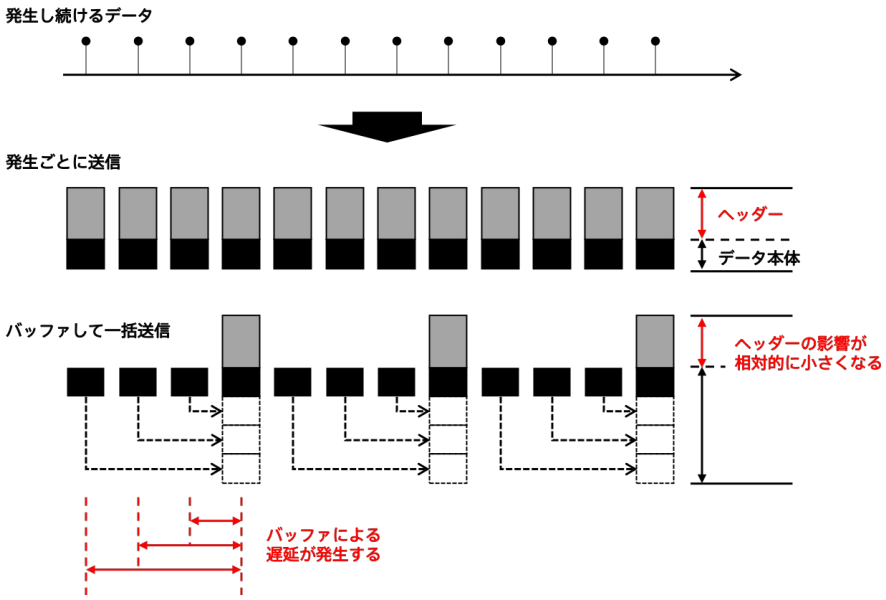
効率伝送

送信バッファリング

小さな高頻度データにそれぞれヘッダを付与して送信することは伝送効率の低下を引き起こします。TCPには、この問題の解決方法としてNagleアルゴリズムという選択肢が提供されていますが、この方法では送信バッファリングによる伝送効率と伝送遅延のトレードオフをきめ細かく設定できません。

伝送効率と伝送遅延に対する要件は、アプリケーション側で想定するユースケースによっても異なります。たとえば、データ収集のようなリアルタイム性に対する要件がさほど厳しくないケースでは、伝送遅延を犠牲にしてでも消費帯域を抑制することがメリットになります。反対に遠隔制御のようなリアルタイム性に厳しい要件が課されるケースでは、バッファリングせずに即時送信することで伝送遅延を最小化することがメリットになります。

上記の例以外にも、さまざまな要件に対して柔軟にトレードオフを設定できるよう、intdashでは、Nagleアルゴリズムを無効化したうえで、送信バッファリングと一括送信の仕組みをユーザーランド上に独自実装しています。

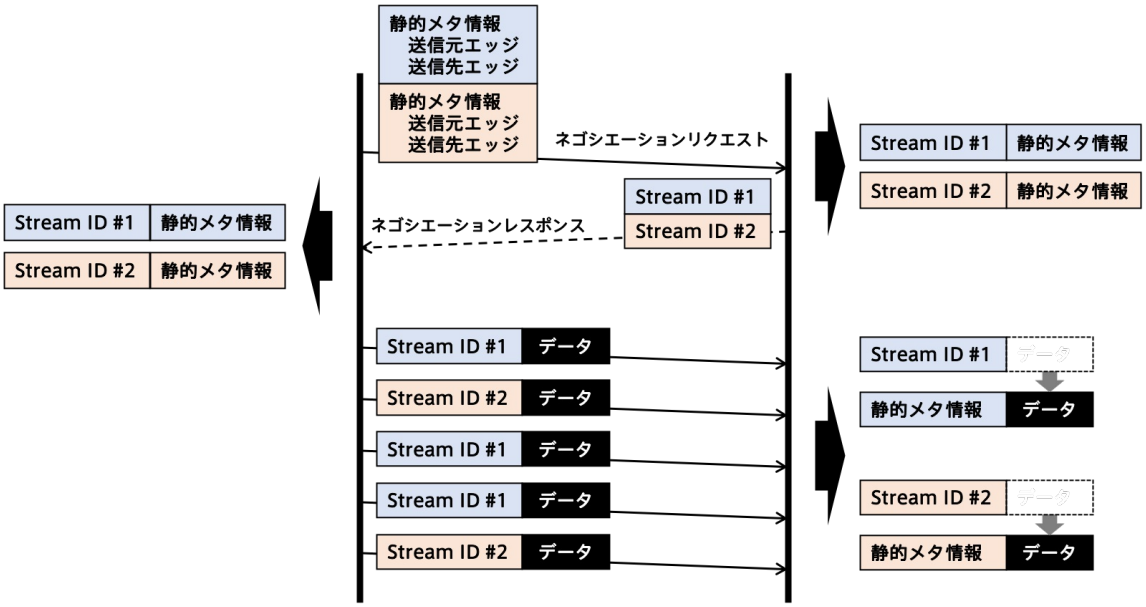


intdashを利用することで、伝送時にどの程度のバッファをもたせるかを要件に応じて設定できるようになります。

冗長性の排除

intdashでは、発生したデータを送信する際に送信元／宛先エッジや送信データの種別、永続化の要否などさまざまなメタ情報が付与されます。高頻度で発生する小さなデータに対し、これらのメタ情報を付与することは、ヘッダ付与によるオーバーヘッド同様、伝送効率の低下を引き起こします。

これを効率低下を回避するため、intdashでは、伝送を開始する前にネゴシエーションというプロセスにより送受信間で静的なメタ情報を共有し合います。この静的なメタ情報を共有したコンテキストをストリームと呼び、伝送開始後はメタ情報の代わりとしてストリームに付けられたIDだけを送信します。



ネゴシエーション：
ストリームの開始時に静的なメタ情報をエッジ・サーバ間で取り決め、一意なIDを払い出します。

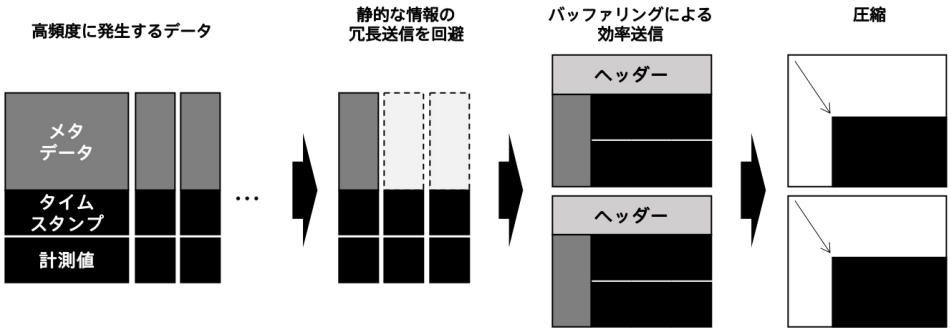
データ送信：
発生したデータを払い出されたIDとともに送信することにより、冗長なメタ情報の送信を回避します。

データ受信：
受信したIDをもとにメタ情報と受信データを対応付けます。

経路圧縮

intdashの現在のバージョンでは、トランスポートプロトコルとしてWebSocketを使用しています。WebSocketを採用する大きな利点のひとつは、経路圧縮機能です。WebSocketは、メッセージごとのDEFLATE圧縮（permessage-deflate）や圧縮コンテキストを利用した圧縮方式拡張（context-takeover）など経路圧縮機能をプロトコルレベルでサポートしており、伝送のリアルタイム性と伝送効率という2点において優れたバランスをもつプロトコルになっています。

intdashでは、前述の送信バッファリングやネゴシエーションにより格納データの無駄を省き、さらにWebSocketの経路圧縮により格納されたデータをさらに小さく圧縮して伝送することで、狭い帯域においても大量データのリアルタイムストリーミングを実現しています。



送信データの流量抑制

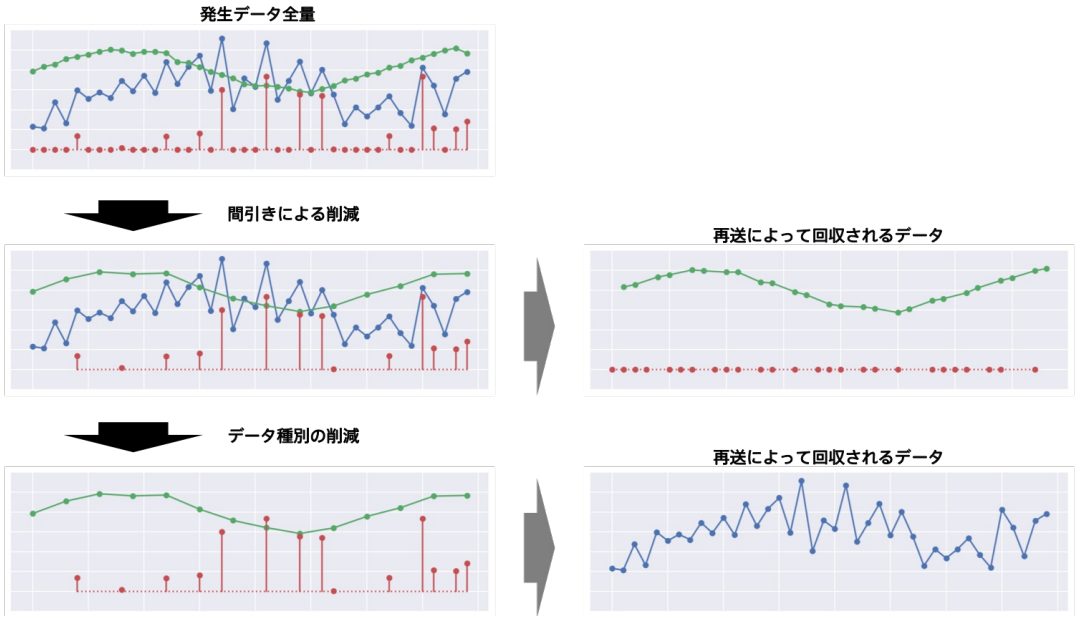
前述のとおり、 intdashは、限られた伝送帯域内で大量のデータを効率的に転送するための機能を備えています。しかし、ネットワークの品質が保証されていない以上、最大限の効率化を図ったとしても発生データ量が伝送帯域に収まらない事態は起こり得ます。このような事態に備えて、intdashでは、事前に設定した優先度にもとづいて低優先度データの送信をあえて再送に回すことにより、優先度の高いデータを送信する帯域を確保します。

再送に回された低優先度データは、リアルタイムデータとして受信側で受け取ることはできませんが、前述の完全回収機能によって確実に再送されるため、永続化されたデータの完全性は担保されます。本機能は、リアルタイムモニタリングで速報値のみを確認しつつ、後の解析用途のためにデータを完全回収しておくようなケースで有用となります。

低優先度データを再送に回す方法については、特定の種類のデータを一度にすべて再送に回したり、特定のサンプリングレートになるまで間引き、間引いた分のみ再送に回すなどさまざまなパターンを組み合わせることができます。

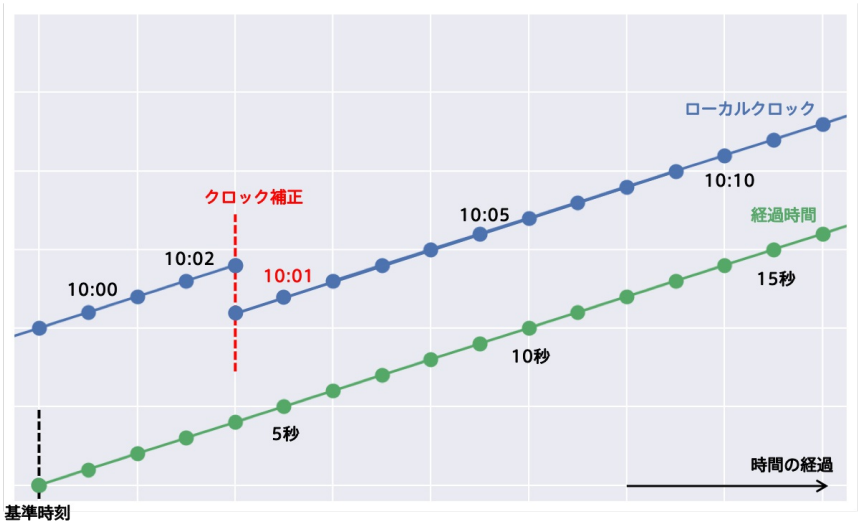
流量抑制の設定例

- 特定のデータの送信を中止する
- 低解像度の動画へ切り替える
- サンプリングレートを下げる



経過時間による時間管理

intdashでは、接続する各デバイスのクロックがずれている場合に備え、データ発生時刻を **ある基準時間から経過した時間** によって管理します。経過時間であれば、クロックが絶対時刻からずれていても、伝送の途中でNTPやGPSによってローカルクロックの時刻が調整されたとしても影響を受けません。



また、経過時間であれば、デバイスがリアルタイムクロックを保有しない場合でも取得できます。このようなデバイスの場合は、別のデバイスのトリガを利用したり、後解析によって手動で設定するなどの方法を取ることにより時刻を合わせることができます。基準時刻と経過時間によって発生時刻が管理されているため、後から時刻を合わせる場合でも、基準となる時刻のみを更新すればよく、各時系列データの時刻をすべて更新する必要はありません。