

モノリスからリアクティブへ

大切なのはアーキテクチャ

James Roper

@jroper



Lightbend

AGENDA

モノリスを置き換える際の落とし穴を明らかにし、リアクティブなソリューションを設計する

Lagom を使ってライブコーディング！

AGENDA

- Identify pitfalls of monolith conversions

モノリスを置き換える際の落とし穴を明らかにし、リアクティブなソリューションを設計する

Lagom を使ってライブコーディング！

AGENDA

- Identify pitfalls of monolith conversions
- Architect reactive solutions

モノリスを置き換える際の落とし穴を明らかにし、リアクティブなソリューションを設計する

Lagom を使ってライブコーディング！

AGENDA

- Identify pitfalls of monolith conversions
- Architect reactive solutions
- See Lagom in action

モノリスを置き換える際の落とし穴を明らかにし、リアクティブなソリューションを設計する

Lagom を使ってライブコーディング！

AGENDA

- Identify pitfalls of monolith conversions
- Architect reactive solutions
- See Lagom in action
- Live coding!

モノリスを置き換える際の落とし穴を明らかにし、リアクティブなソリューションを設計する

Lagom を使ってライブコーディング！

LAGOM AUCTION

Lagom でモノリシックなオークションサイトをマイクロサービスに変換

めざせ eBay !

LAGOM AUCTION

- ebay clone

Lagom でモノリシックなオークションサイトをマイクロサービスに変換

めざせ eBay !

LAGOM AUCTION

- ebay clone
- Was a monolith, converted to microservices

Lagom でモノリシックなオークションサイトをマイクロサービスに変換

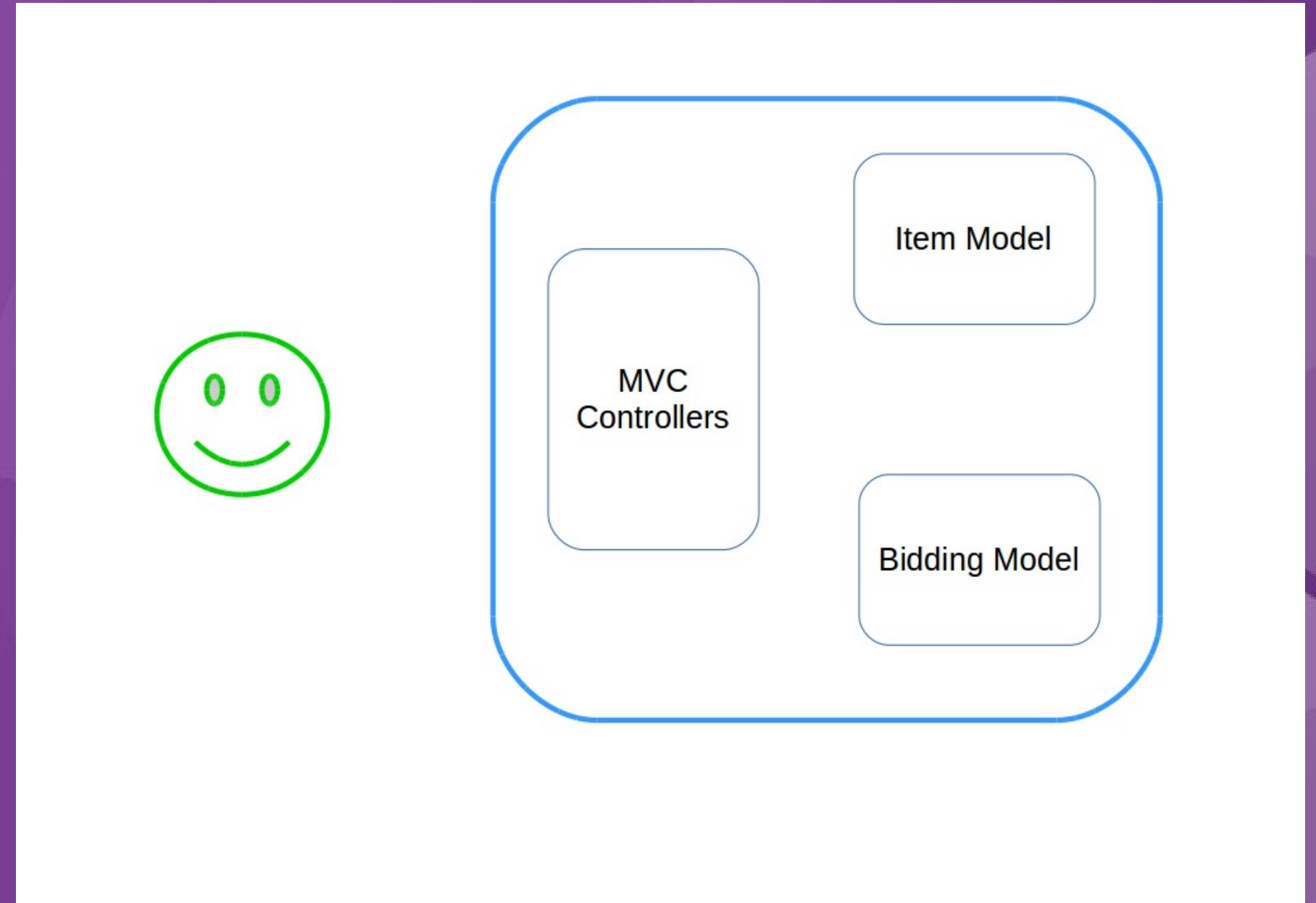
めざせ eBay !

LAGOM AUCTION

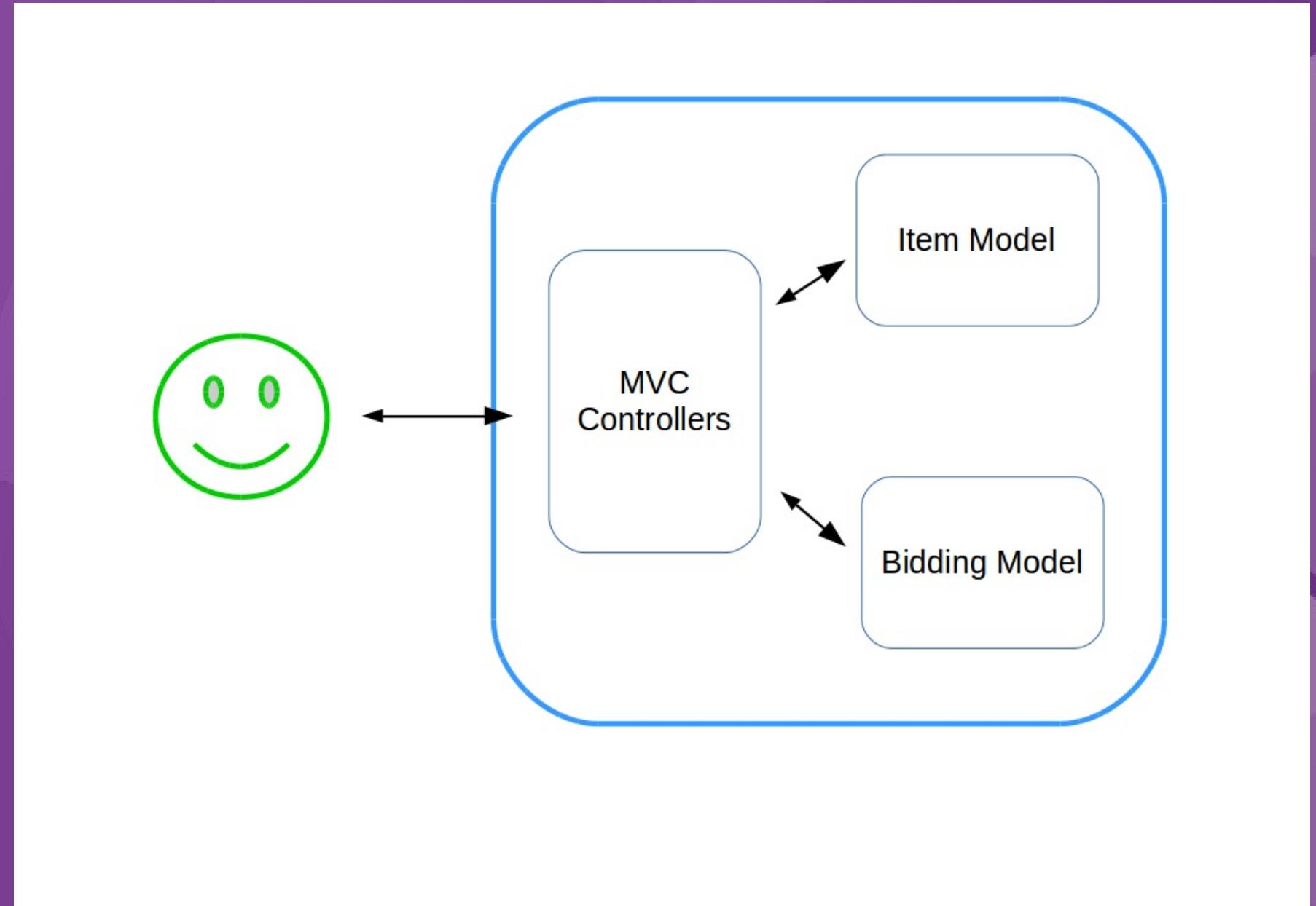
- ebay clone
- Was a monolith, converted to microservices
- Will one day overtake ebay!

Lagom でモノリシックなオークションサイトをマイクロサービスに変換

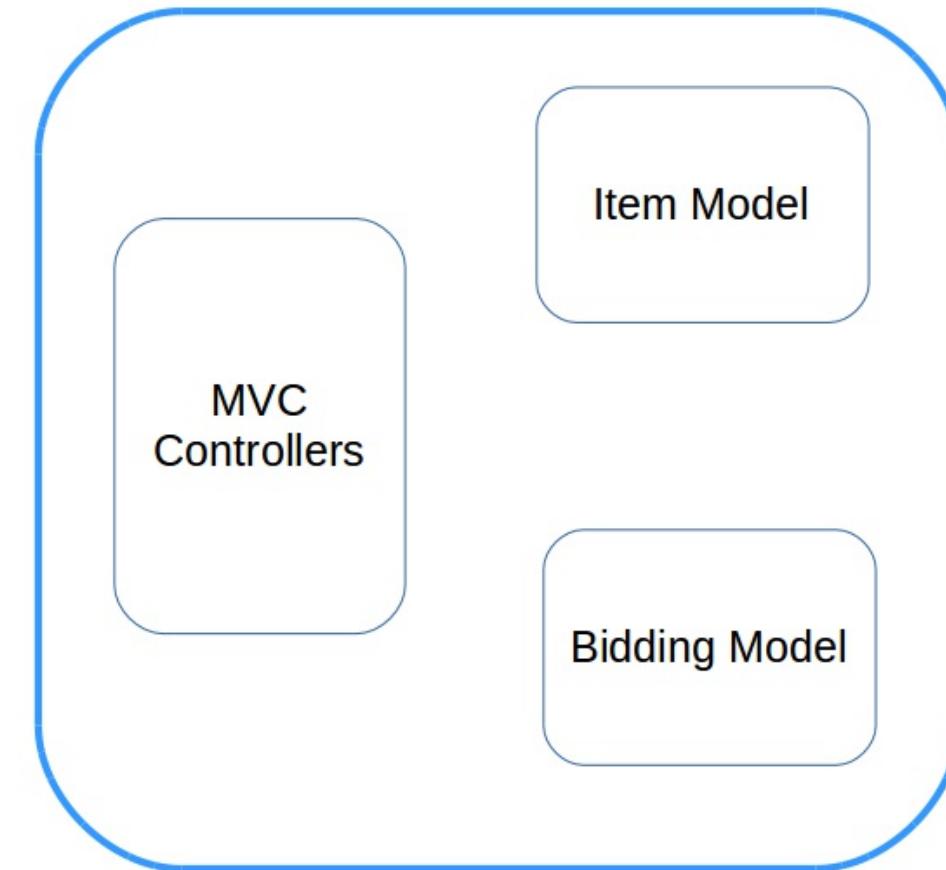
めざせ eBay !



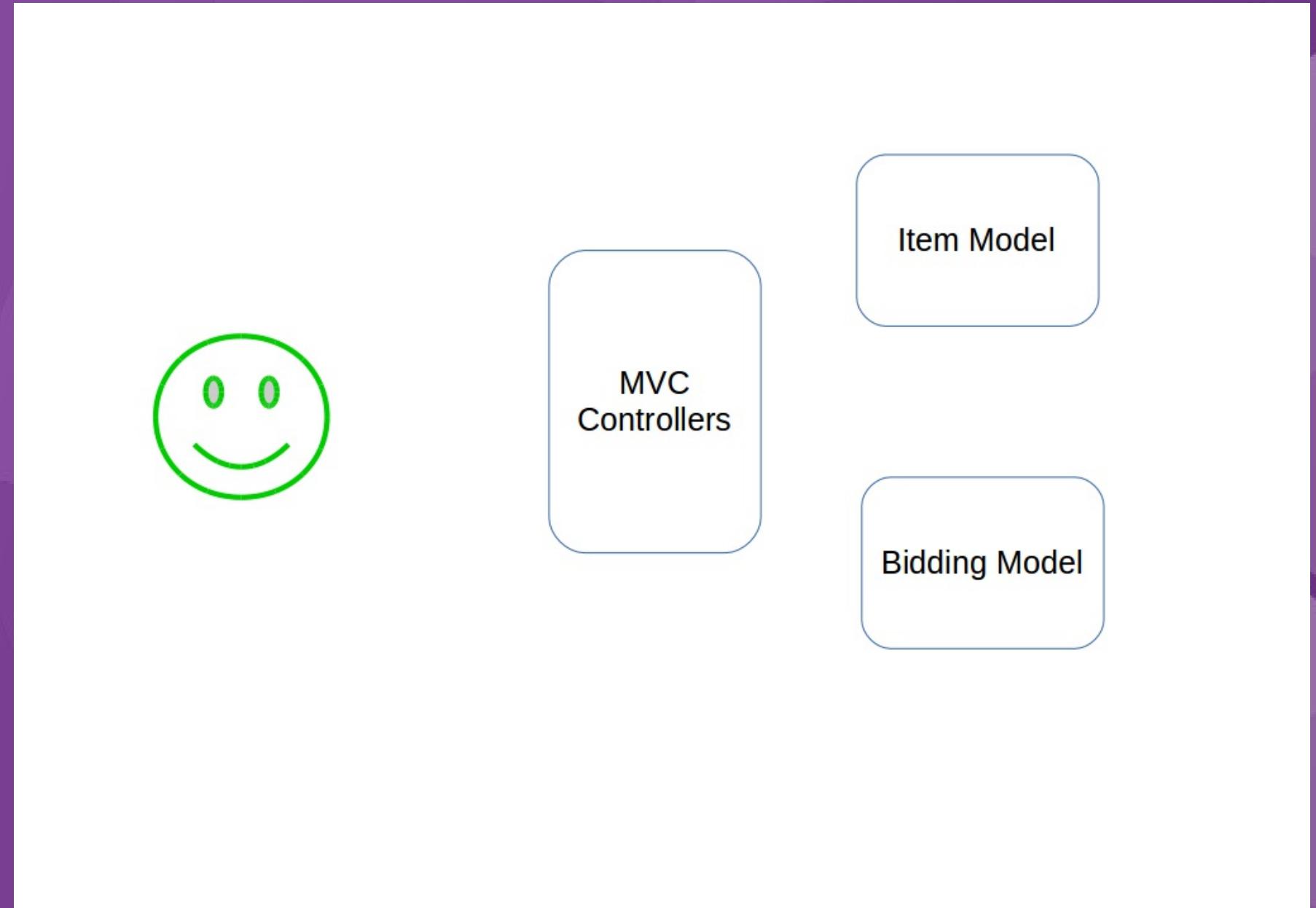
モノリスから商品サービスと入札サービスを切り出してゲートウェイを置く



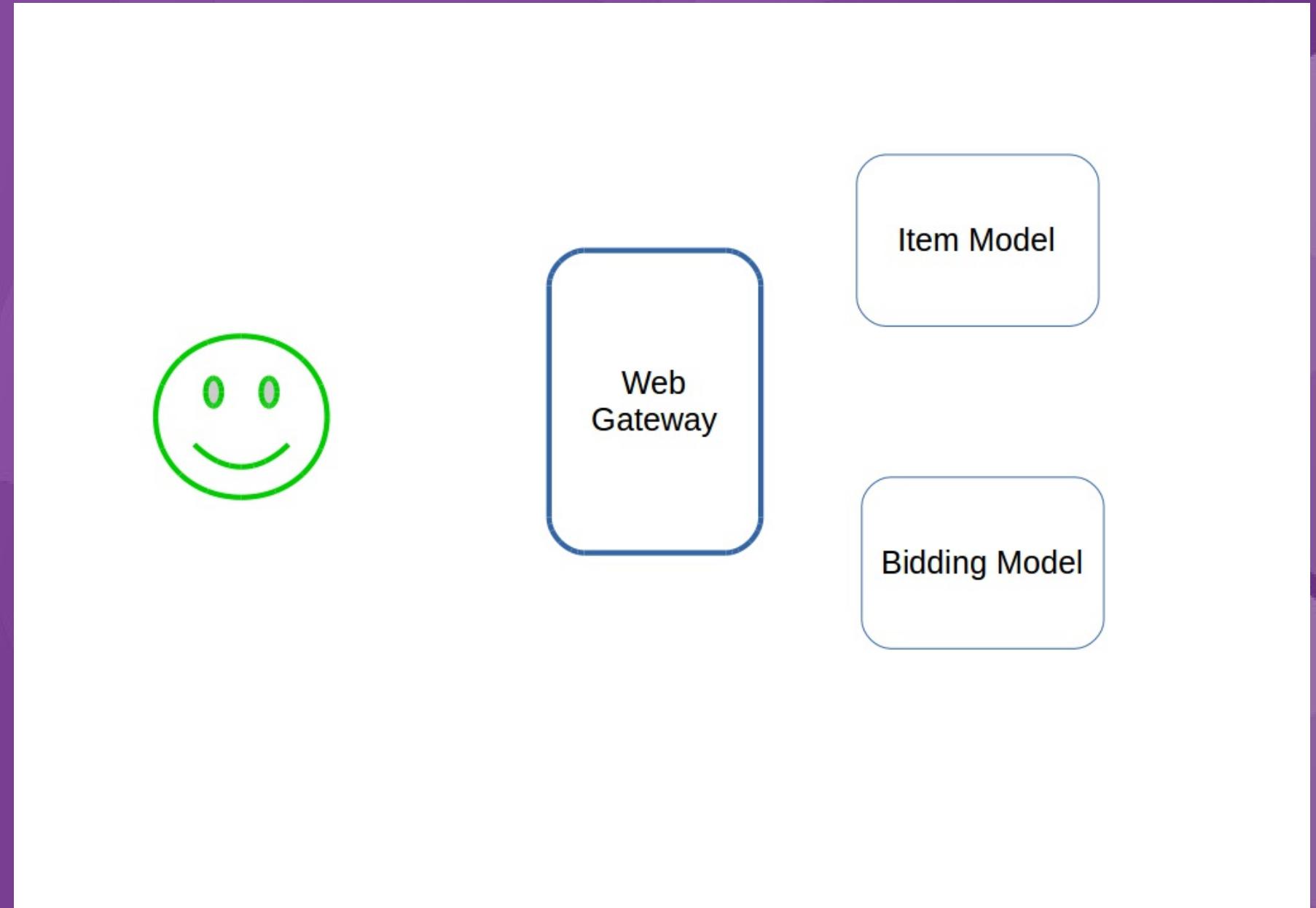
モノリスから商品サービスと入札サービスを切り出してゲートウェイを置く



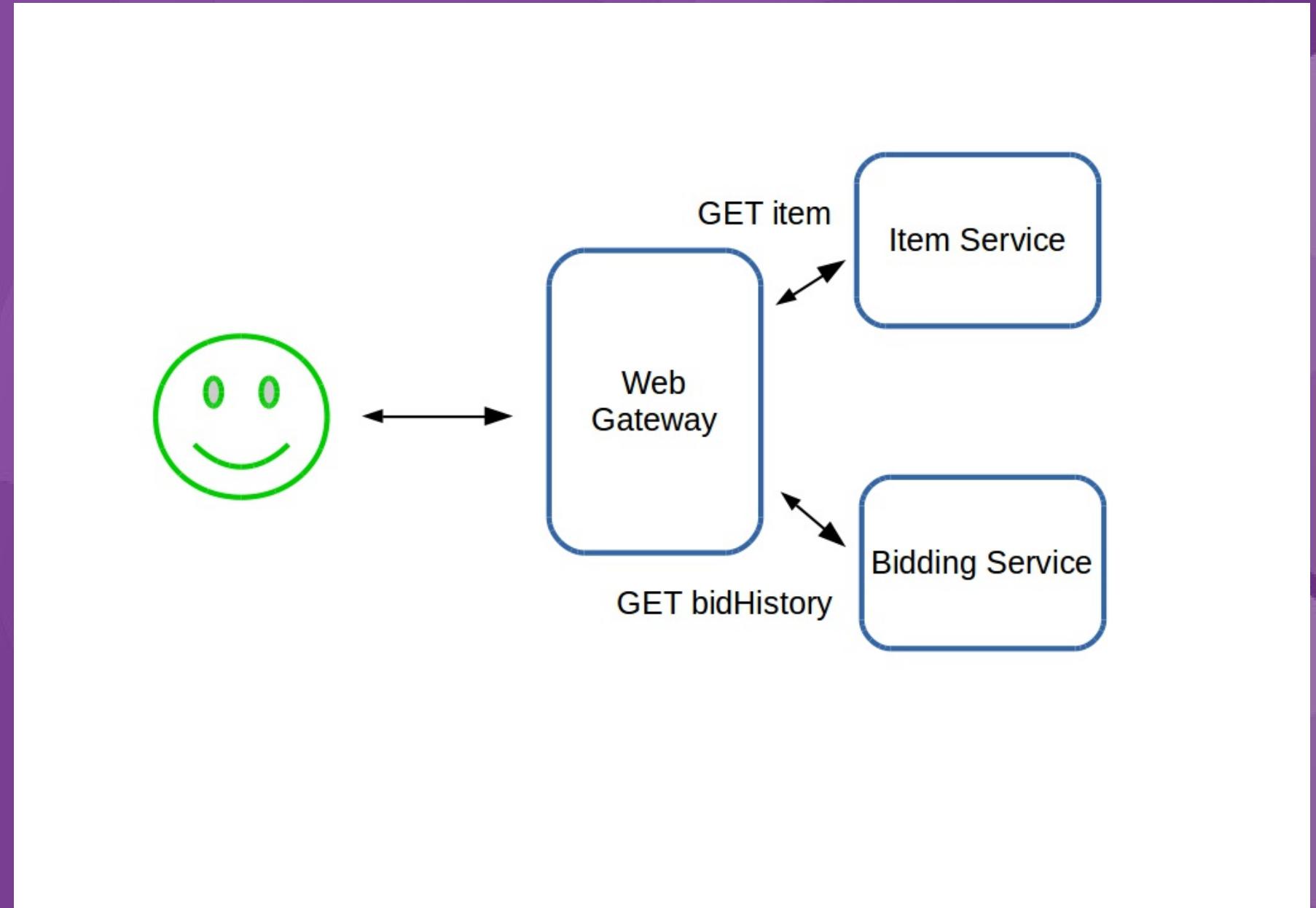
モノリスから商品サービスと入札サービスを切り出してゲートウェイを
置く



モノリスから商品サービスと入札サービスを切り出してゲートウェイを置く



モノリスから商品サービスと入札サービスを切り出してゲートウェイを置く



モノリスから商品サービスと入札サービスを切り出してゲートウェイを置く

WHAT IF SOMETHING GOES WRONG?

マイクロサービス化すると可動部分が増える

それによって障害が増え、一貫性が崩れやすくなる

WHAT IF SOMETHING GOES WRONG?

- Microservices means more moving parts

マイクロサービス化すると可動部分が増える

それによって障害が増え、一貫性が崩れやすくなる

WHAT IF SOMETHING GOES WRONG?

- Microservices means more moving parts
 - More chance for failure

マイクロサービス化すると可動部分が増える

それによって障害が増え、一貫性が崩れやすくなる

WHAT IF SOMETHING GOES WRONG?

- Microservices means more moving parts
 - More chance for failure
 - More chance for inconsistency

マイクロサービス化すると可動部分が増える

それによって障害が増え、一貫性が崩れやすくなる

SYNCHRONOUS COMMUNICATION

同期性 - 物事が同時に存在したり起きたりすること

SYNCHRONOUS COMMUNICATION

synchronous *adj.* - existing or occurring at the same time.

同期性 - 物事が同時に存在したり起きたりすること

SYNCHRONOUS COMMUNICATION

リクエストする側とレスポンスする側が、同時に即応性を保つ必要がある

SYNCHRONOUS COMMUNICATION

- Typically request/response

リクエストする側とレスポンスする側が、同時に即応性を保つ必要がある

SYNCHRONOUS COMMUNICATION

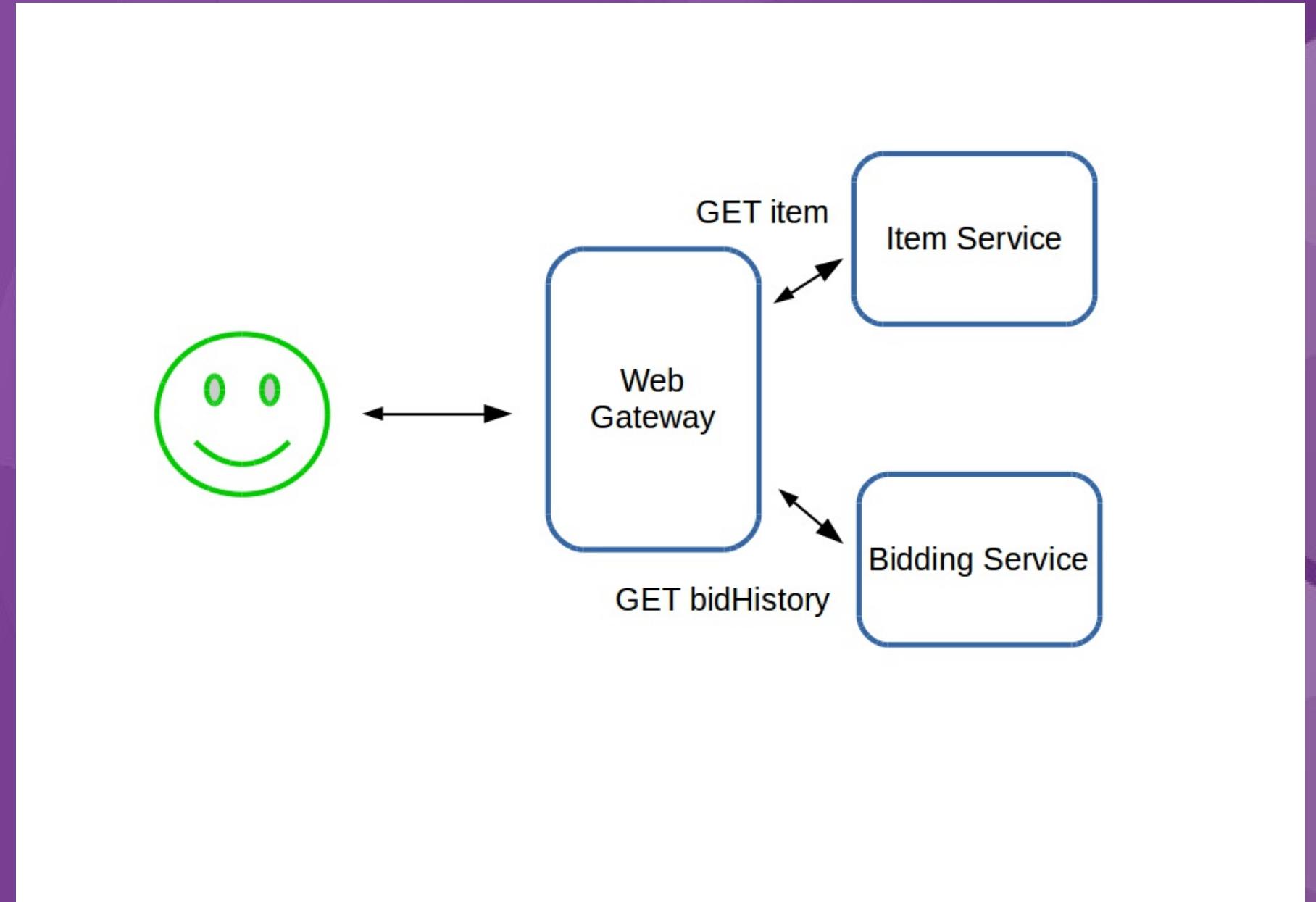
- Typically request/response
 - e.g. REST

リクエストする側とレスポンスする側が、同時に即応性を保つ必要がある

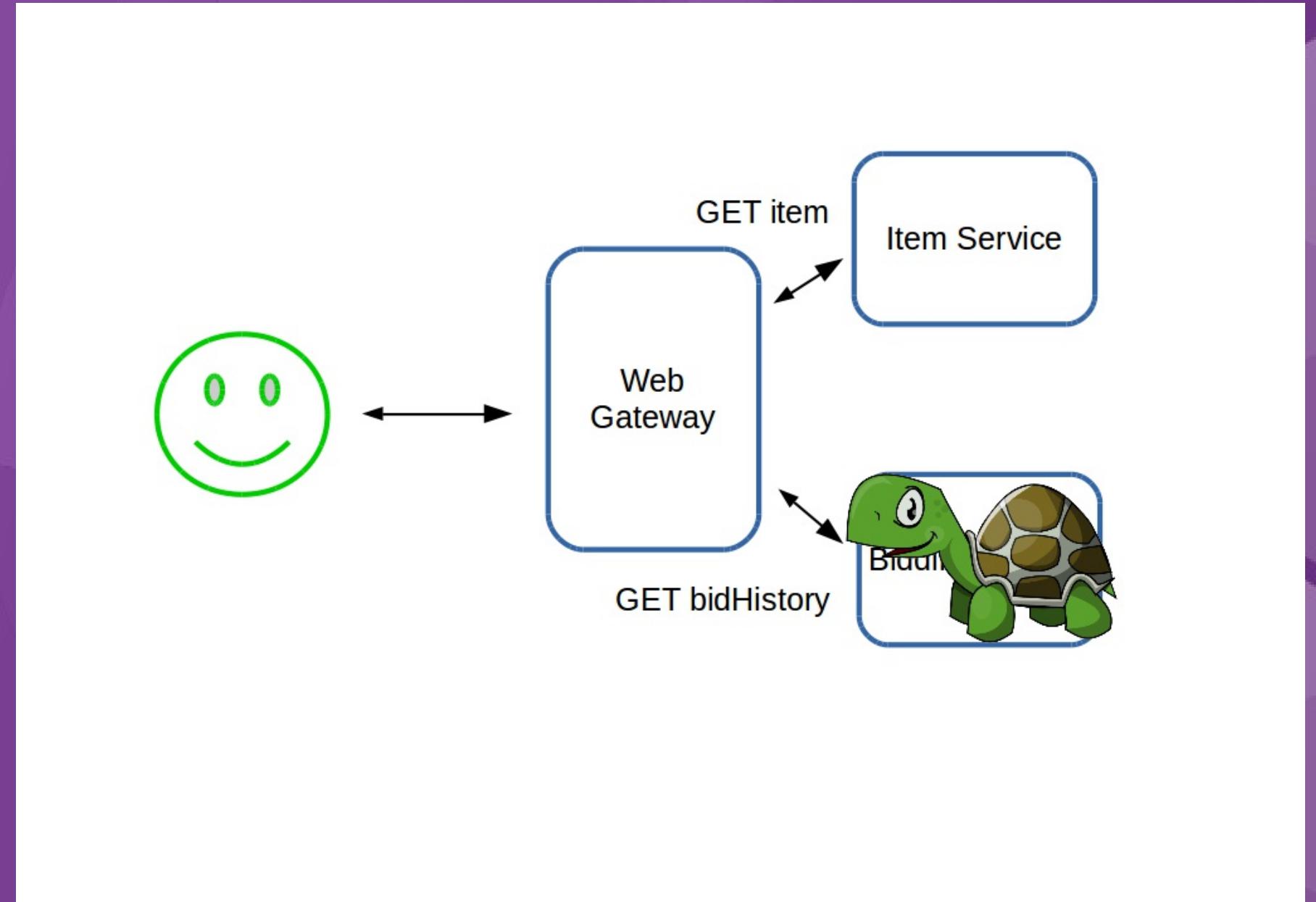
SYNCHRONOUS COMMUNICATION

- Typically request/response
 - e.g. REST
- Both systems must be responsive at the same time

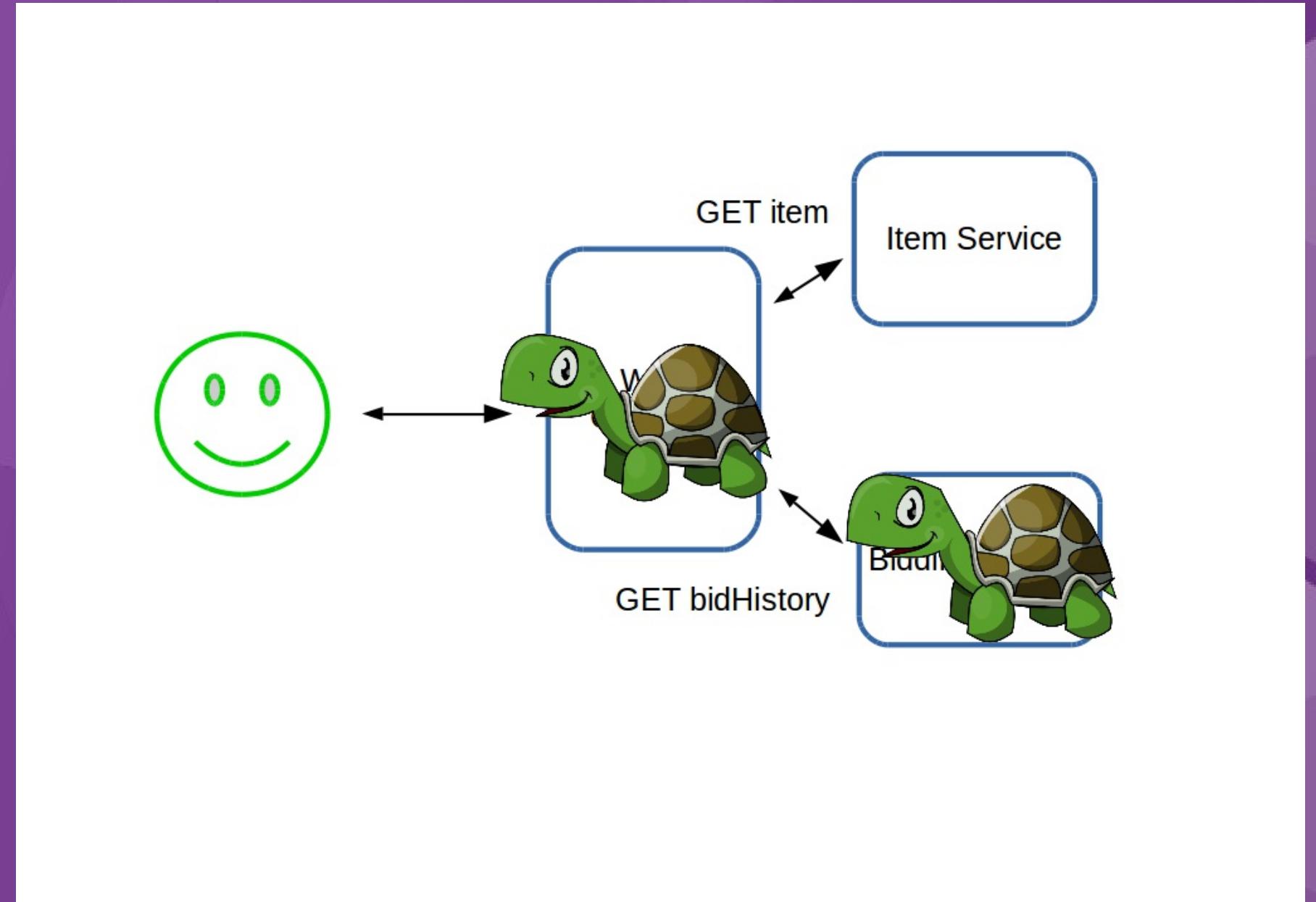
リクエストする側とレスポンスする側が、同時に即応性を保つ必要がある



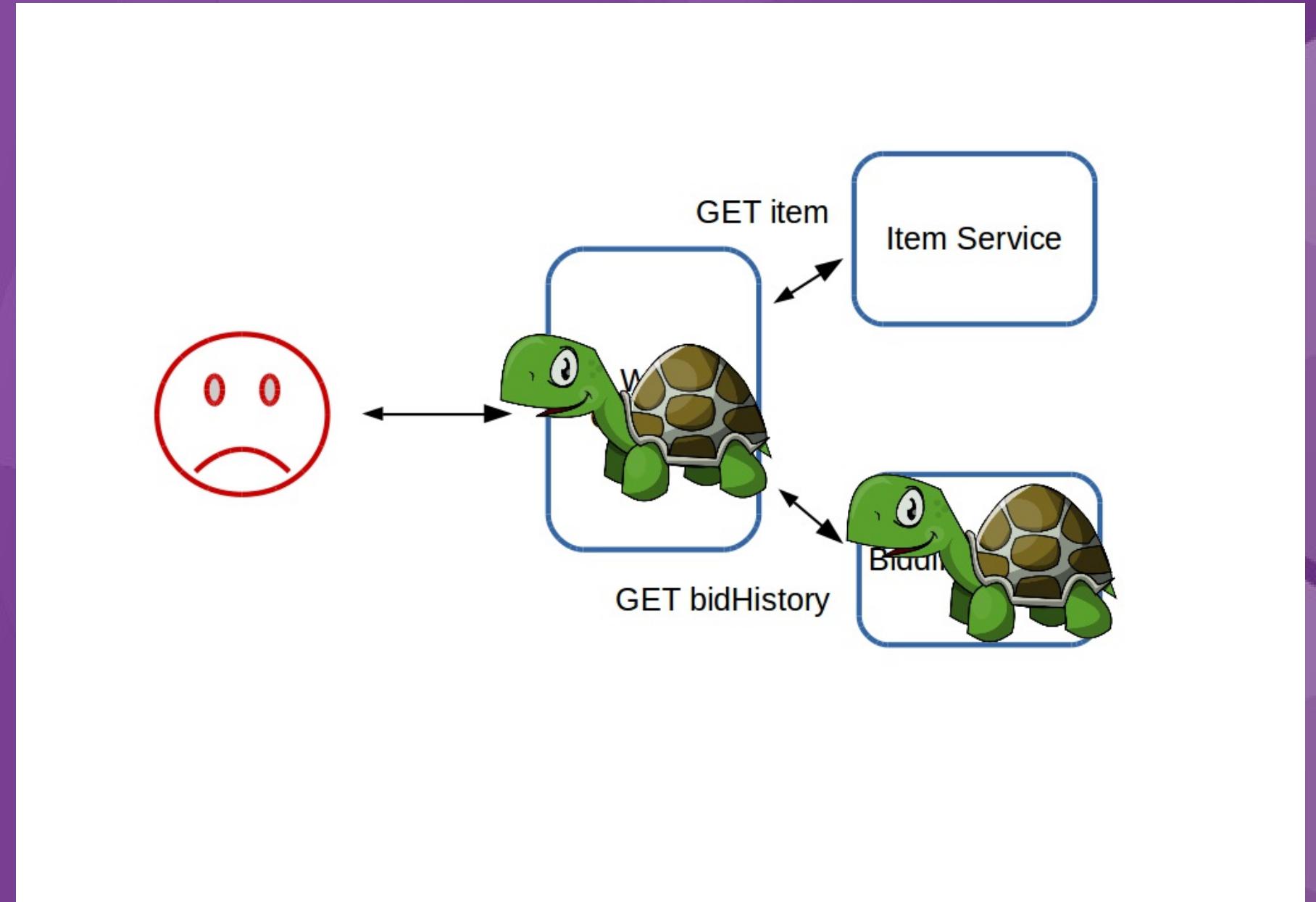
入札サービスが遅延すると、商品サービスが正常でも遅延が発生する



入札サービスが遅延すると、商品サービスが正常でも遅延が発生する



入札サービスが遅延すると、商品サービスが正常でも遅延が発生する



入札サービスが遅延すると、商品サービスが正常でも遅延が発生する

PATTERN 1: CIRCUIT BREAKERS

サーキットブレーカー: 障害時にブレーカーを上げて、障害を起こした
サービスを保護する

PATTERN 1: CIRCUIT BREAKERS

- A gate that opens in the event of failure

サーキットブレーカー: 障害時にブレーカーを上げて、障害を起こした
サービスを保護する

PATTERN 1: CIRCUIT BREAKERS

- A gate that opens in the event of failure
 - Including timeouts

サーキットブレーカー: 障害時にブレーカーを上げて、障害を起こした
サービスを保護する

PATTERN 1: CIRCUIT BREAKERS

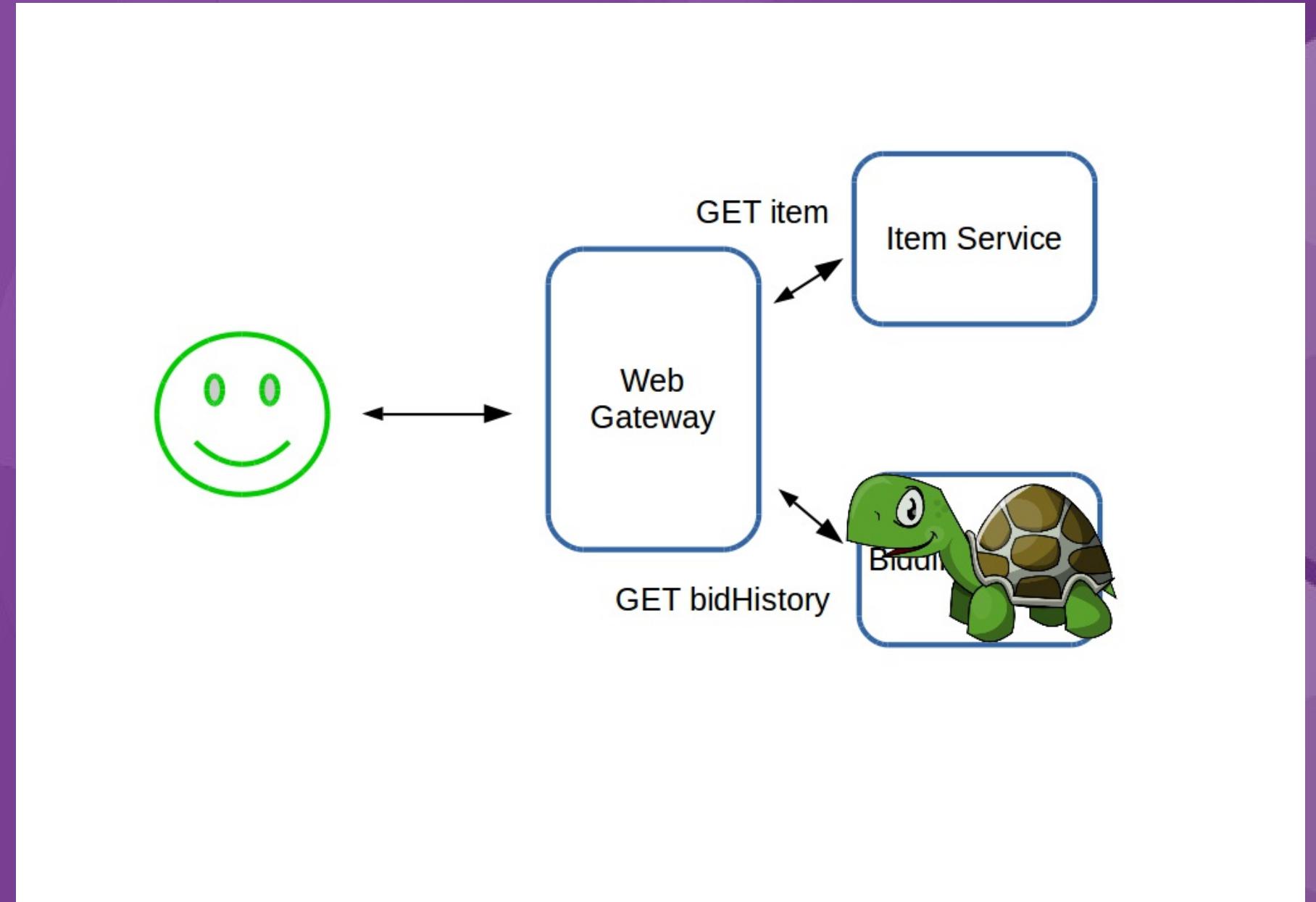
- A gate that opens in the event of failure
 - Including timeouts
- Protects already failing services

サーキットブレーカー: 障害時にブレーカーを上げて、障害を起こした
サービスを保護する

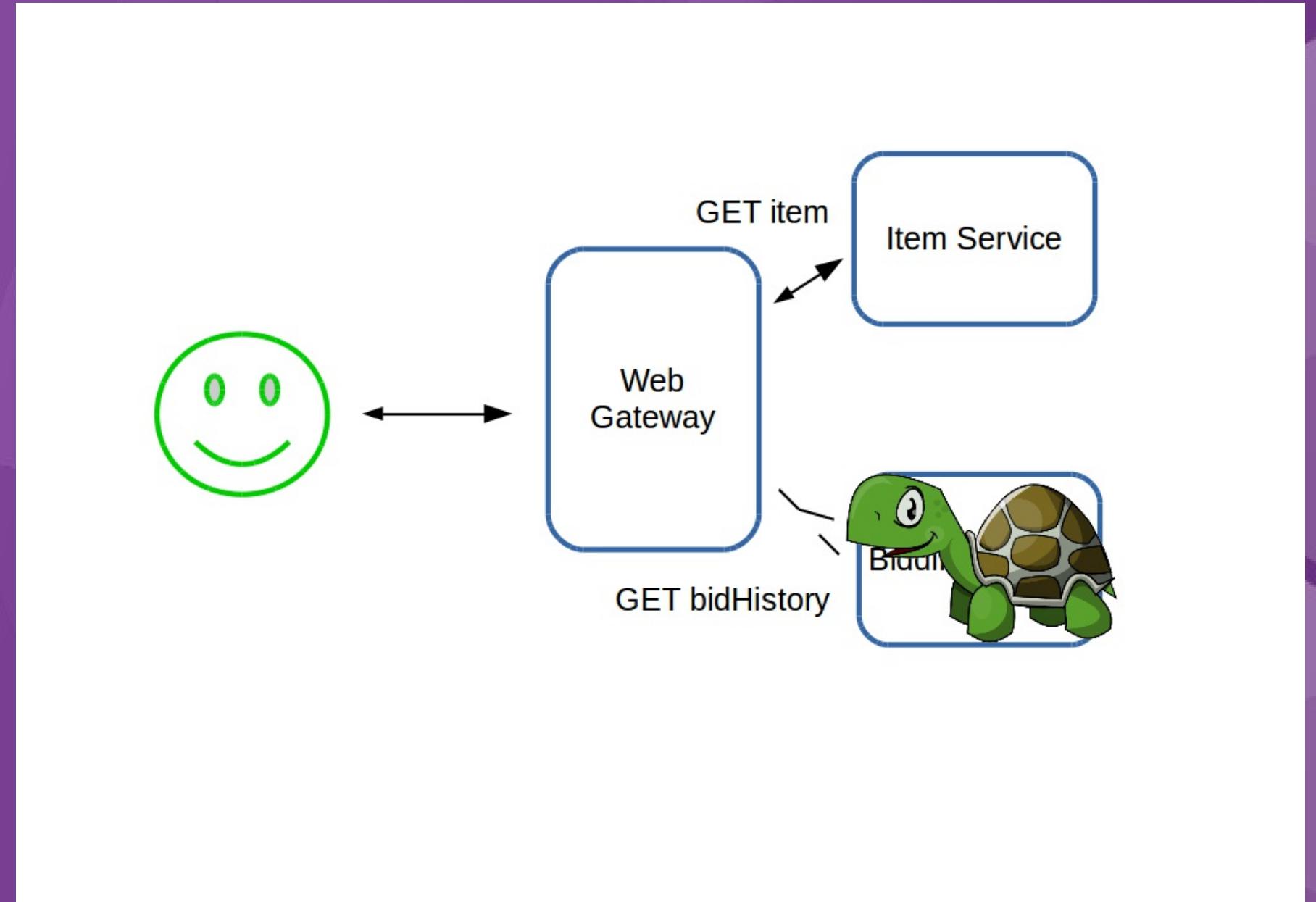
PATTERN 1: CIRCUIT BREAKERS

- A gate that opens in the event of failure
 - Including timeouts
- Protects already failing services
- Allows fail fast handling

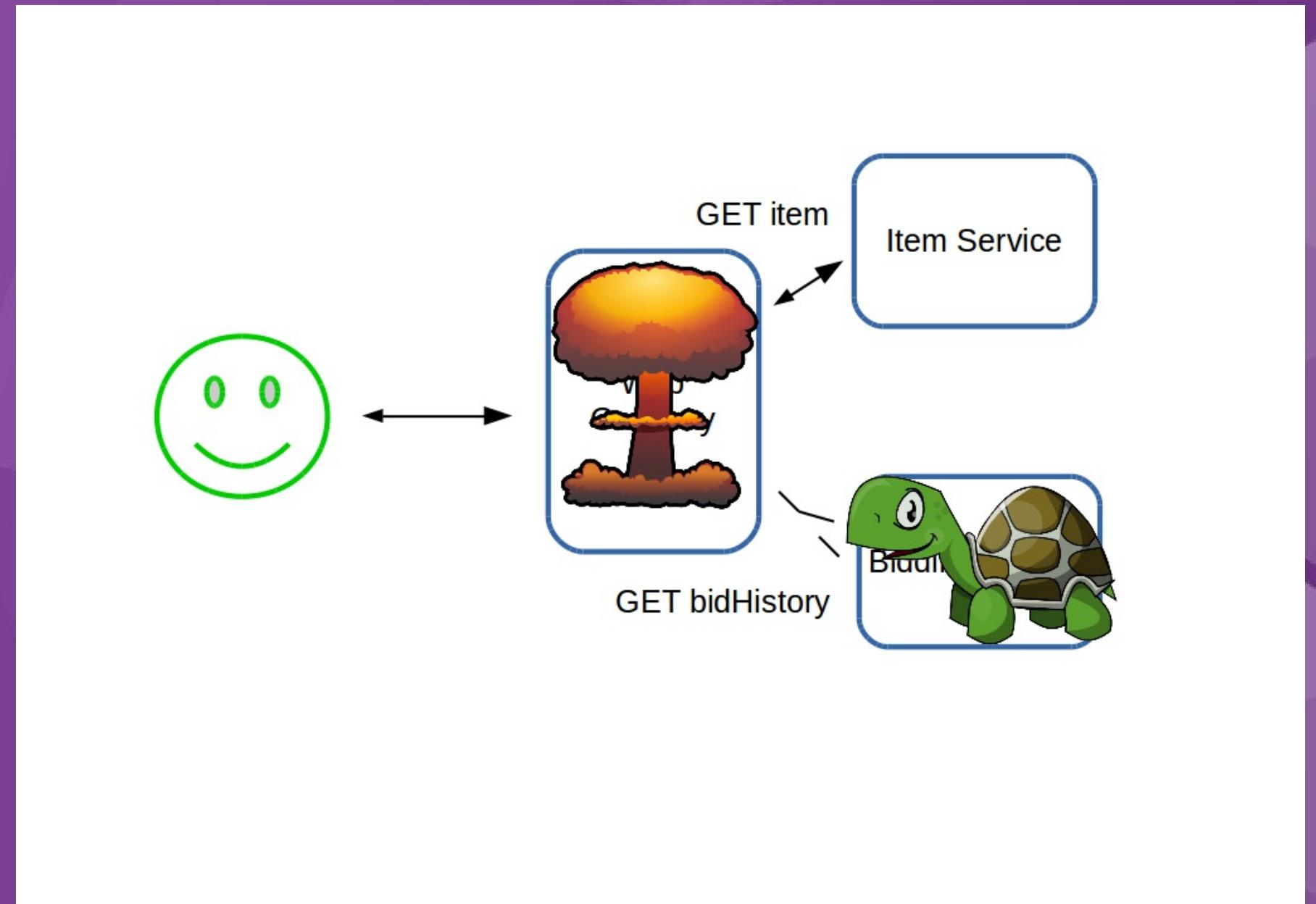
サーキットブレーカー: 障害時にブレーカーを上げて、障害を起こした
サービスを保護する



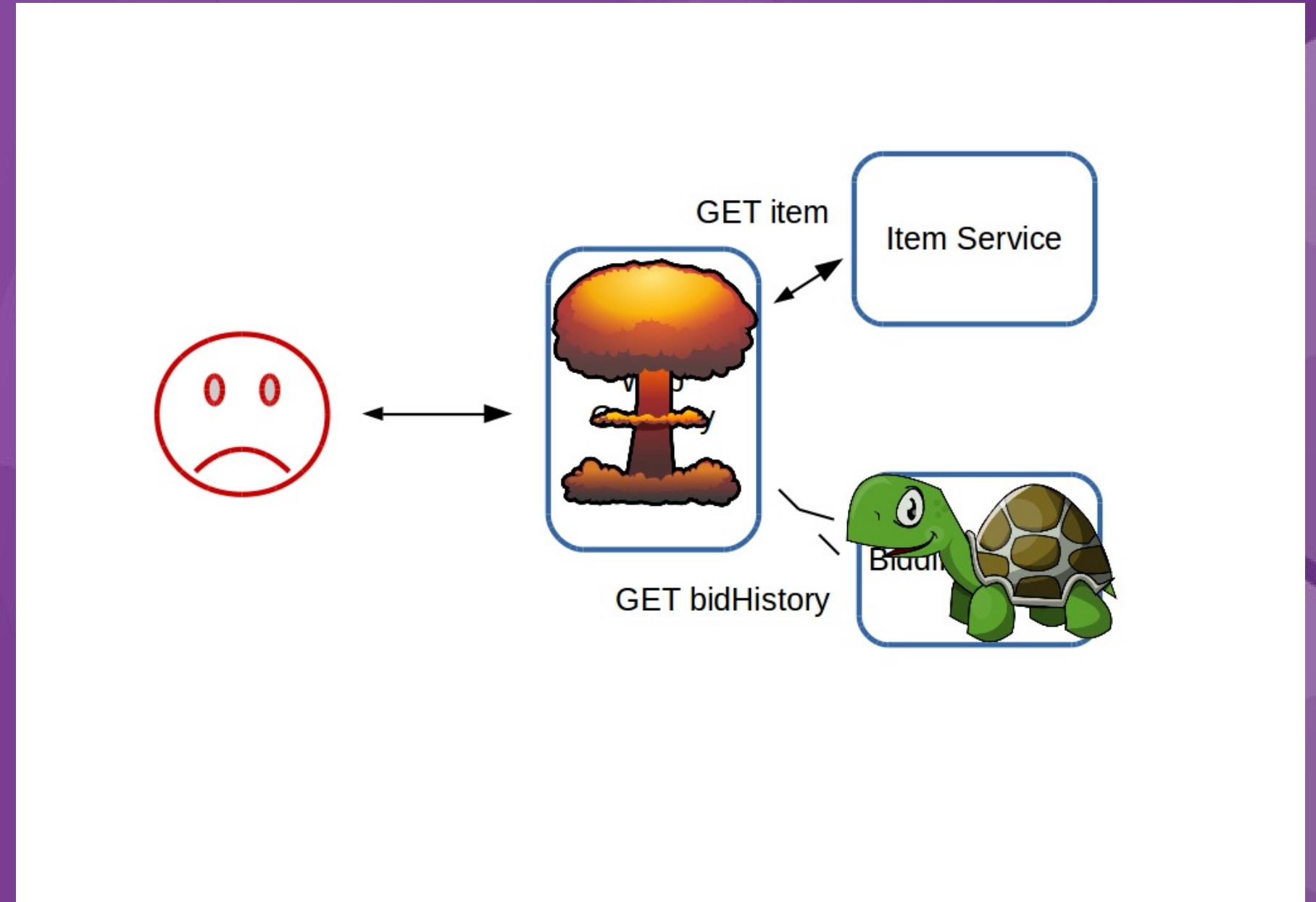
ゲートウェイは、入札サービスの遅延を検出してブレーカーを上げる



ゲートウェイは、入札サービスの遅延を検出してブレーカーを上げる



ゲートウェイは、入札サービスの遅延を検出してブレーカーを上げる



ゲートウェイは、入札サービスの遅延を検出してブレーカーを上げる

PATTERN 2: FAILURE RECOVERY

障害復旧: サービス品質を下げる代わりに障害をワークアラウンドする

各ページのレンダリングに全てのマイクロサービス呼び出しが必要なわけではない

PATTERN 2: FAILURE RECOVERY

- Work around failure by degrading

障害復旧: サービス品質を下げて障害をワークアラウンドする

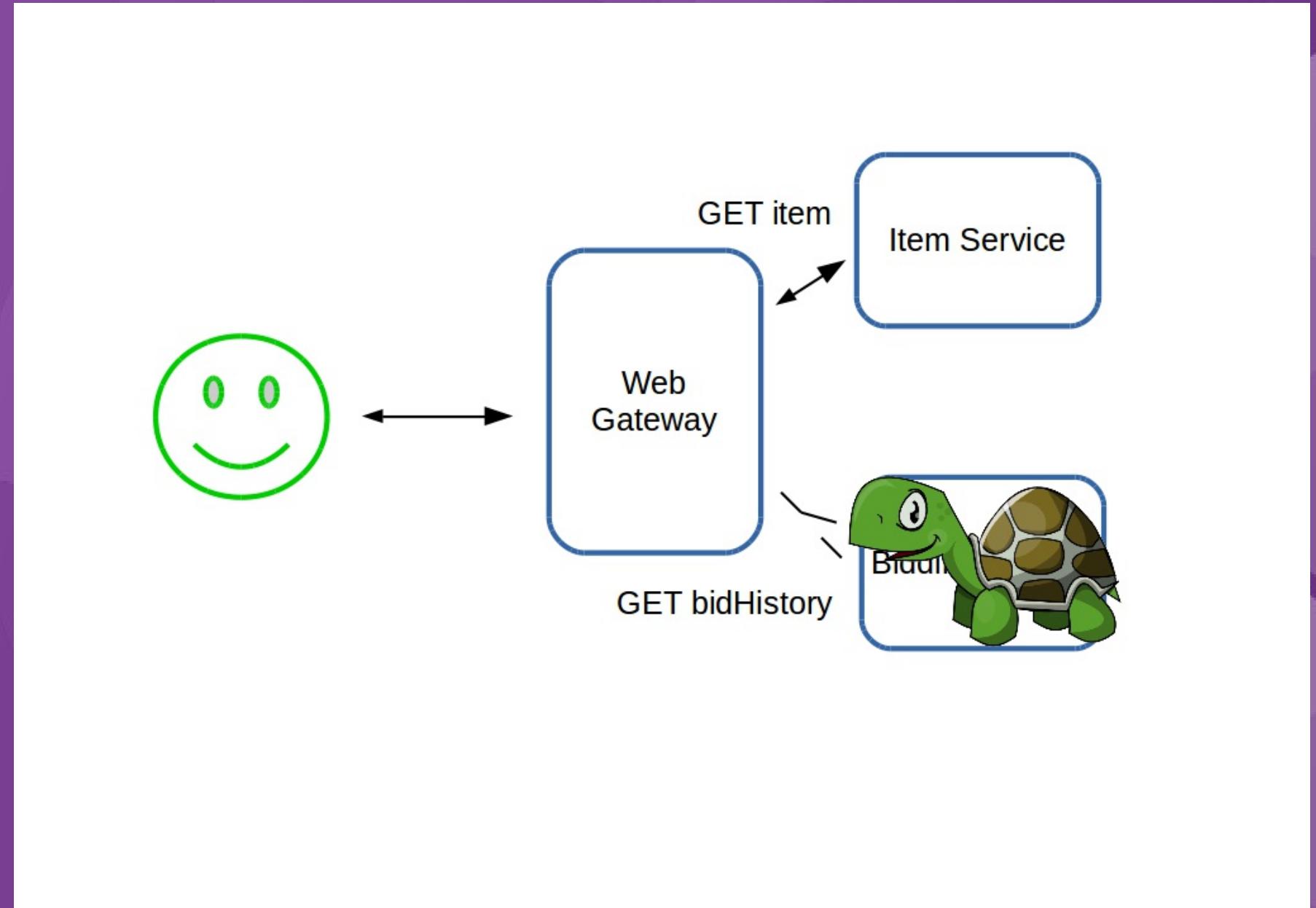
各ページのレンダリングに全てのマイクロサービス呼び出しが必要なわけではない

PATTERN 2: FAILURE RECOVERY

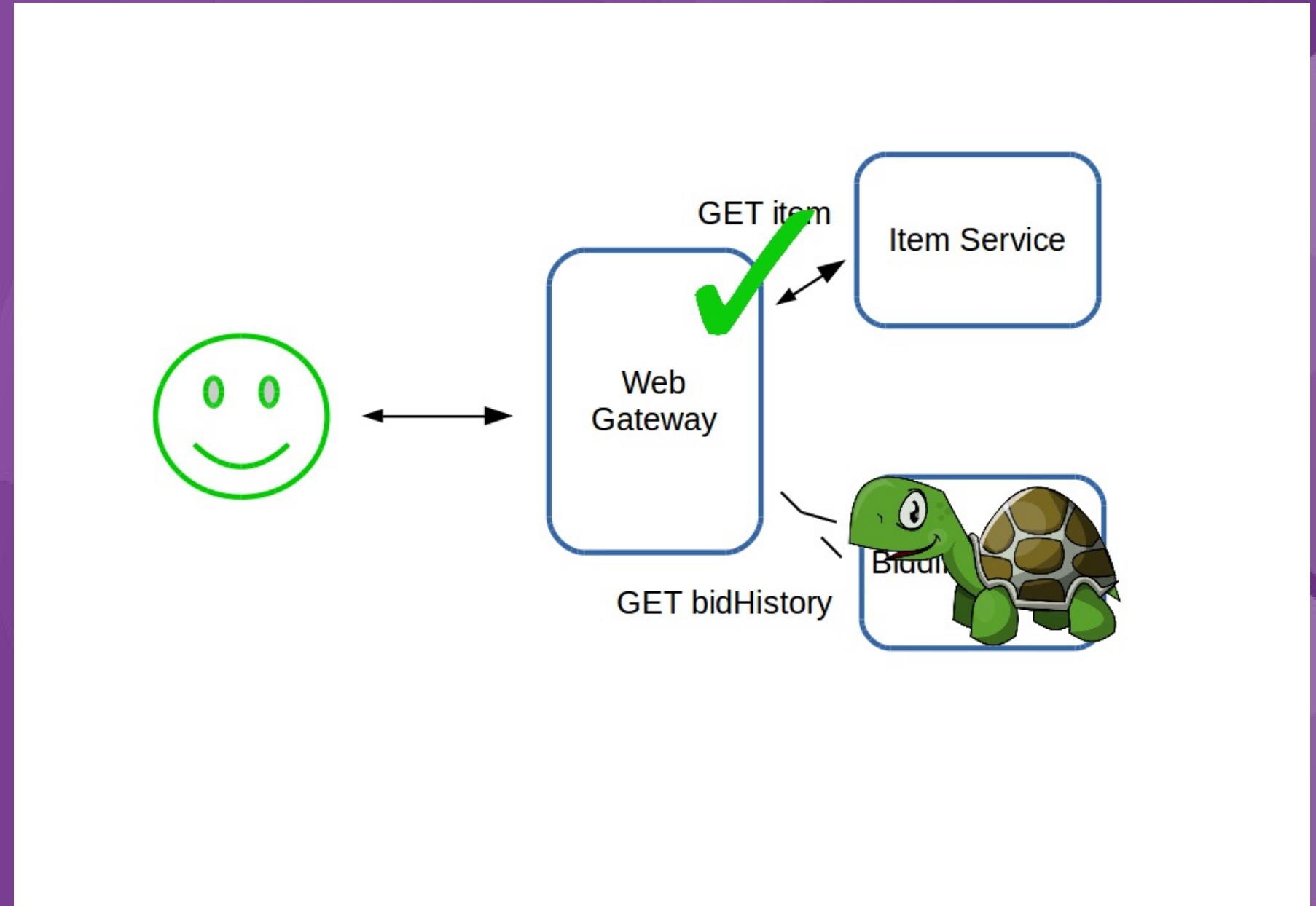
- Work around failure by degrading
- Not every call is necessary to render every page

障害復旧: サービス品質を下げて障害をワークアラウンドする

各ページのレンダリングに全てのマイクロサービス呼び出しが必要なわけではない



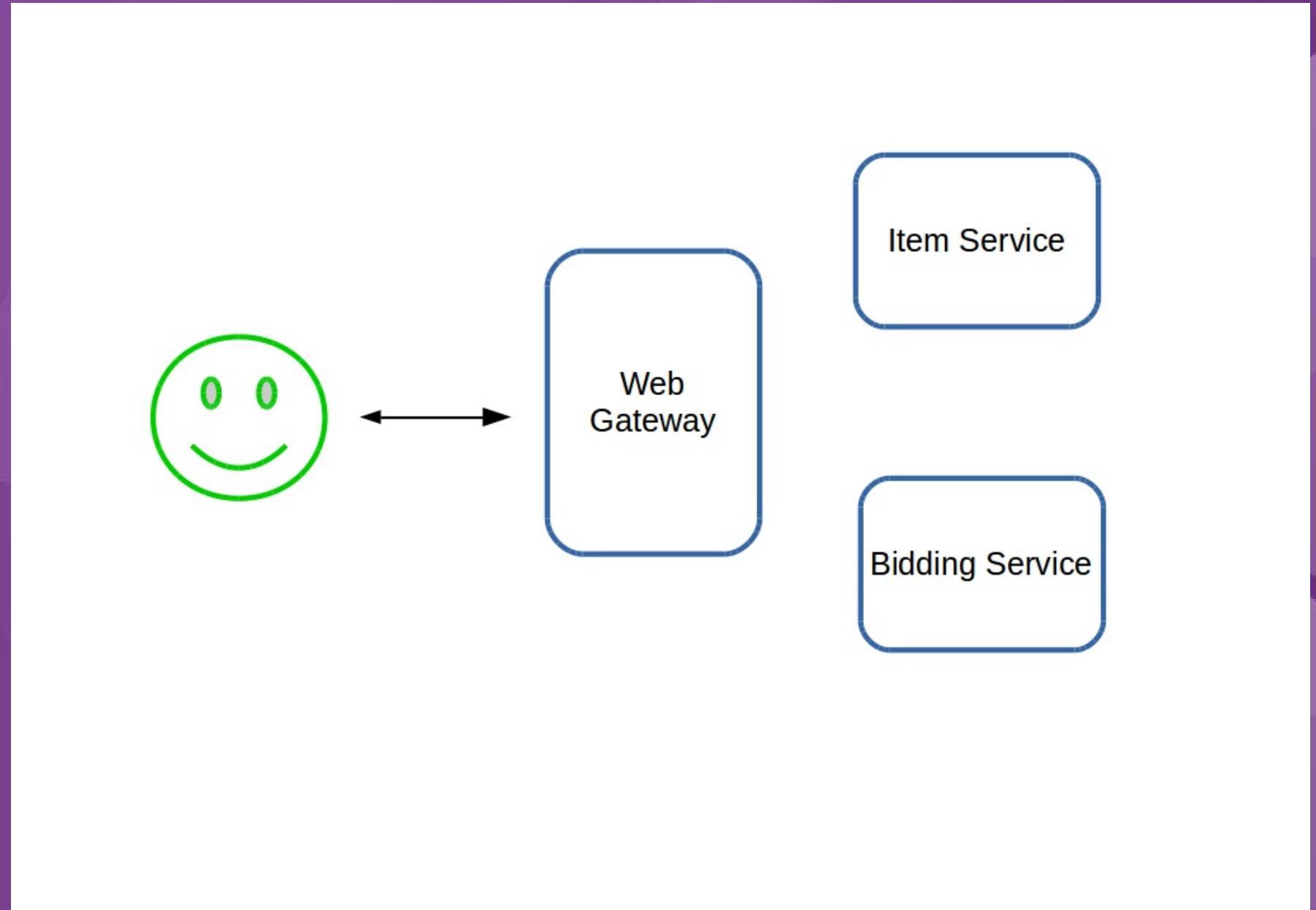
商品サービスが動いているから、オークションサイト全体は止めずに済む



商品サービスが動いているから、オークションサイト全体は止めずに済む

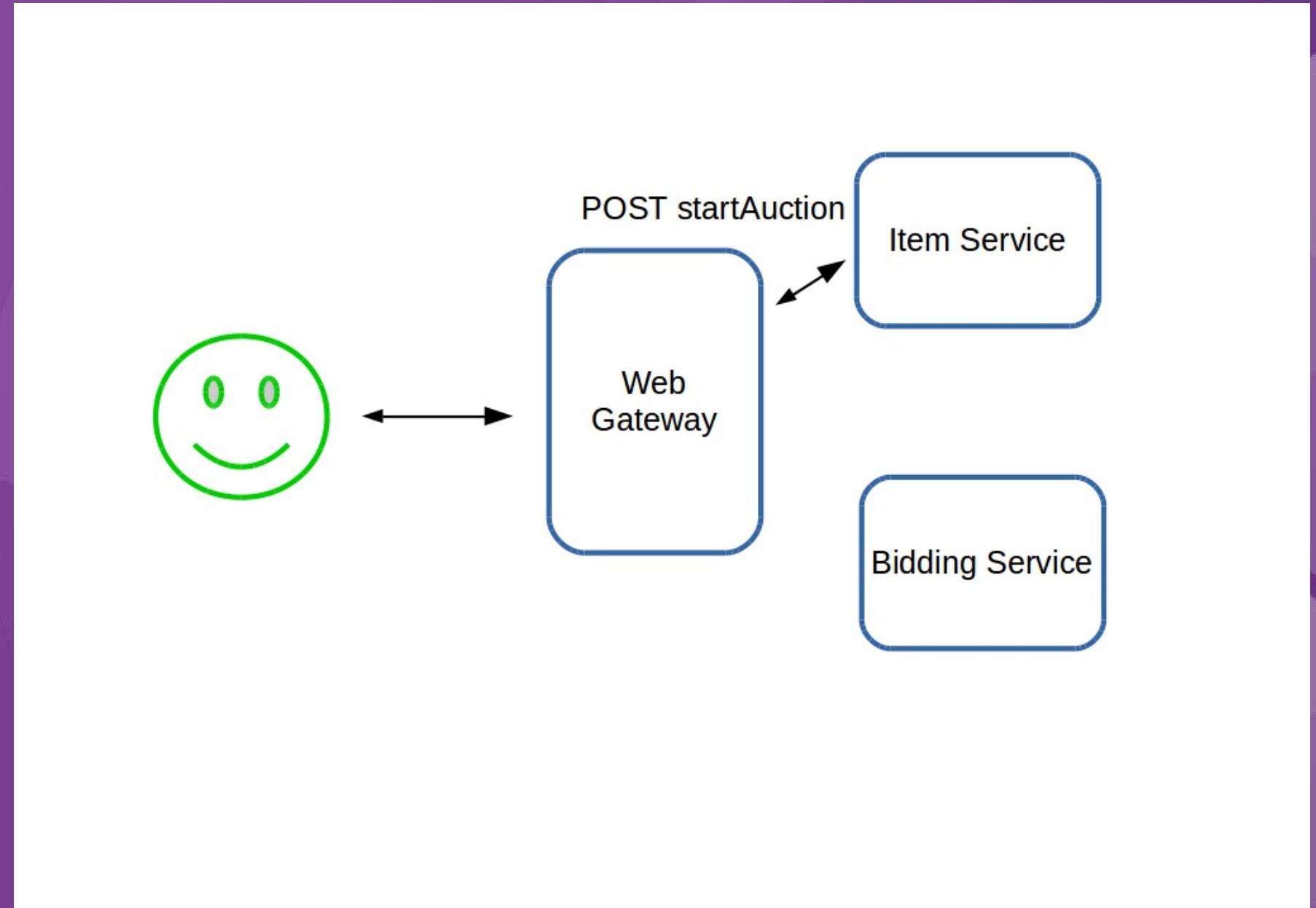


FAILURE CAN LEAD TO
INCONSISTENCY



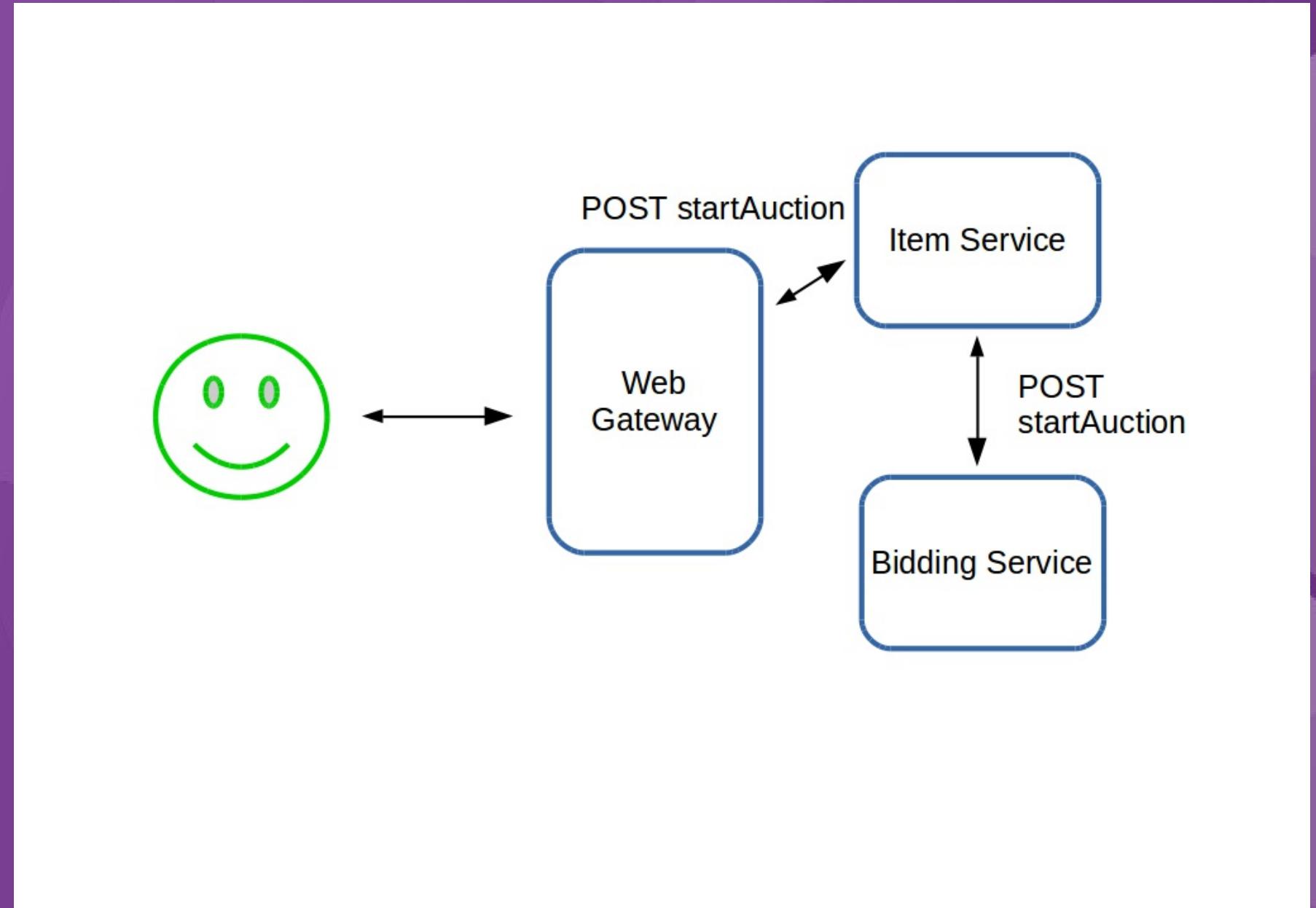
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



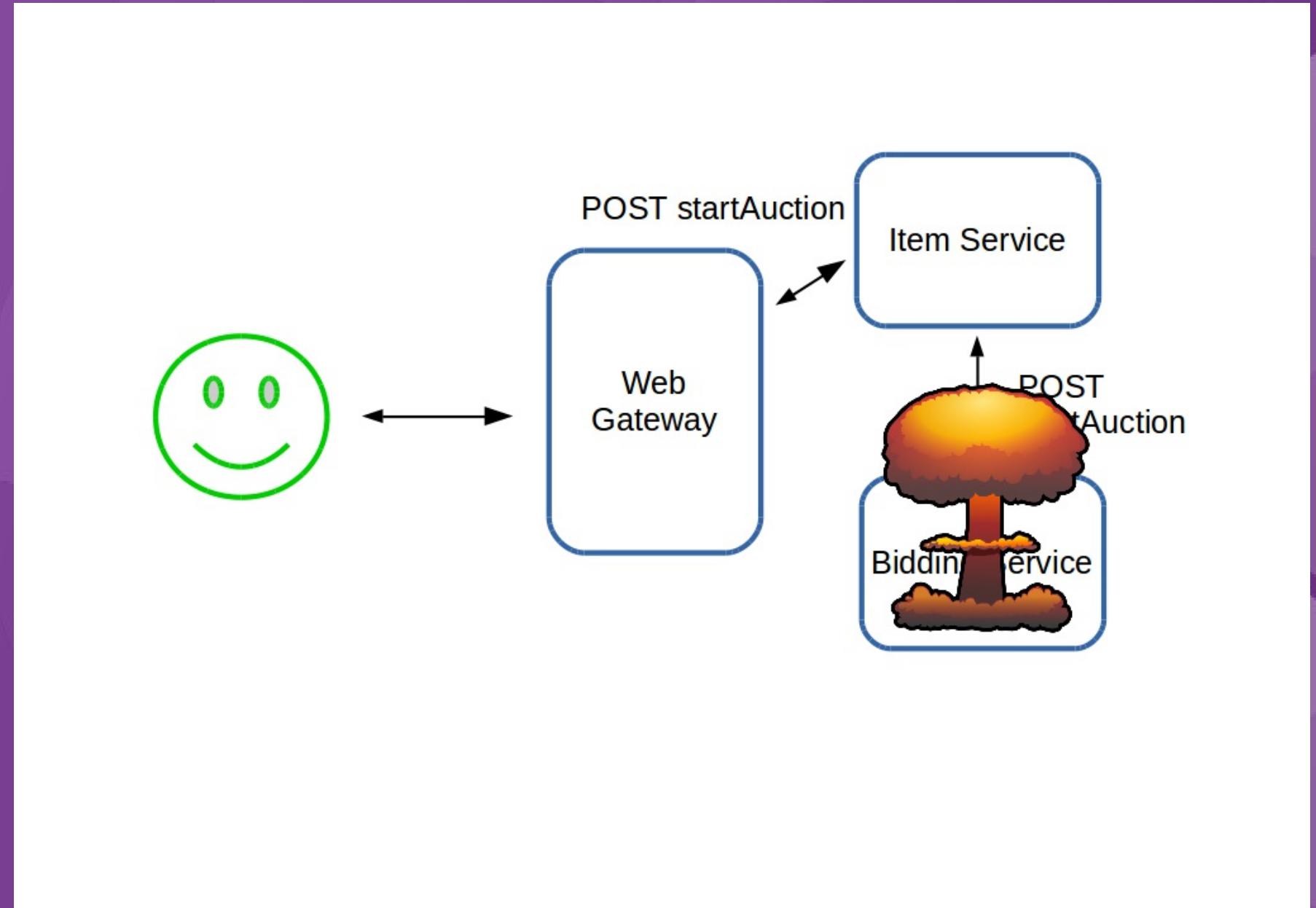
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



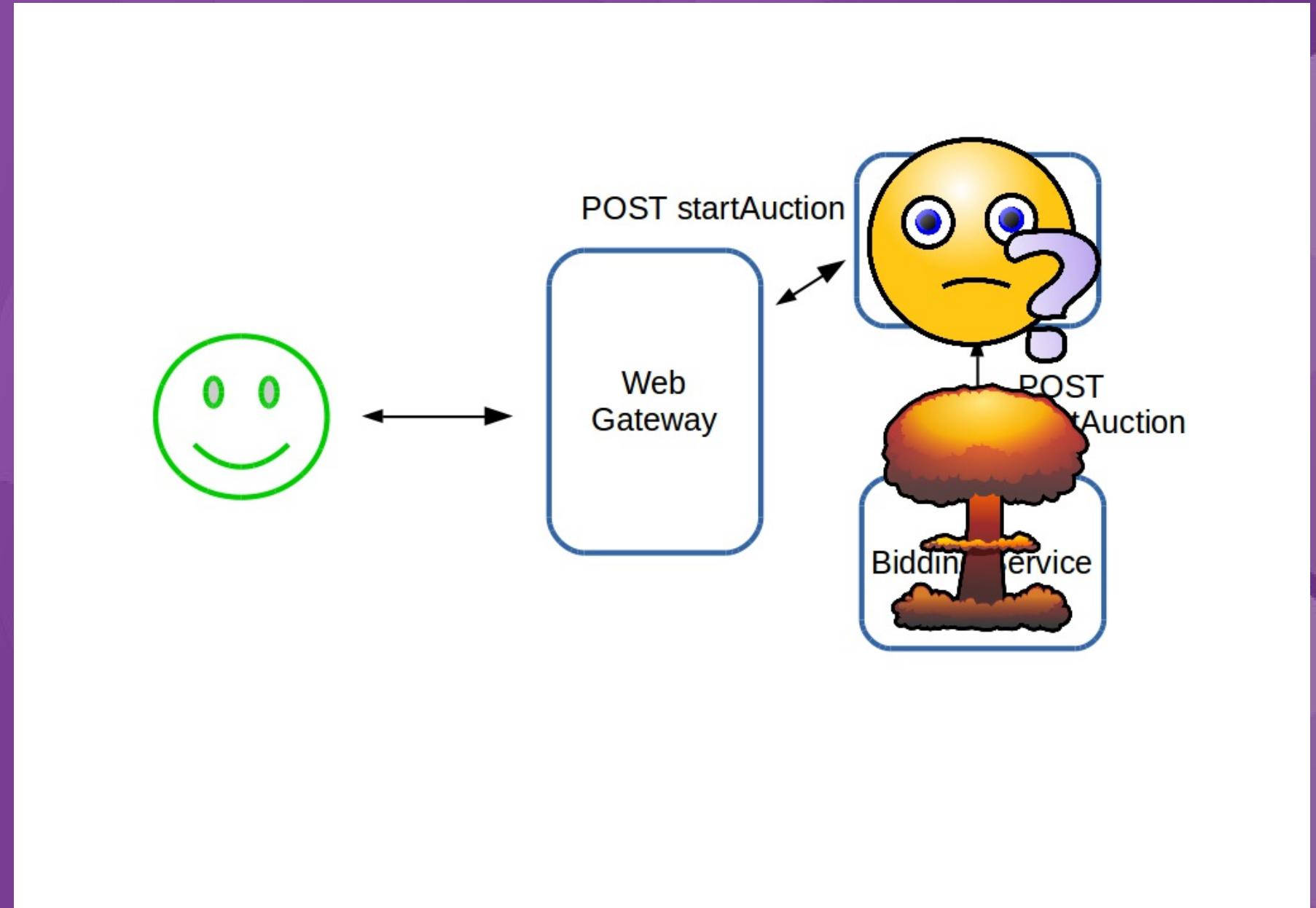
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



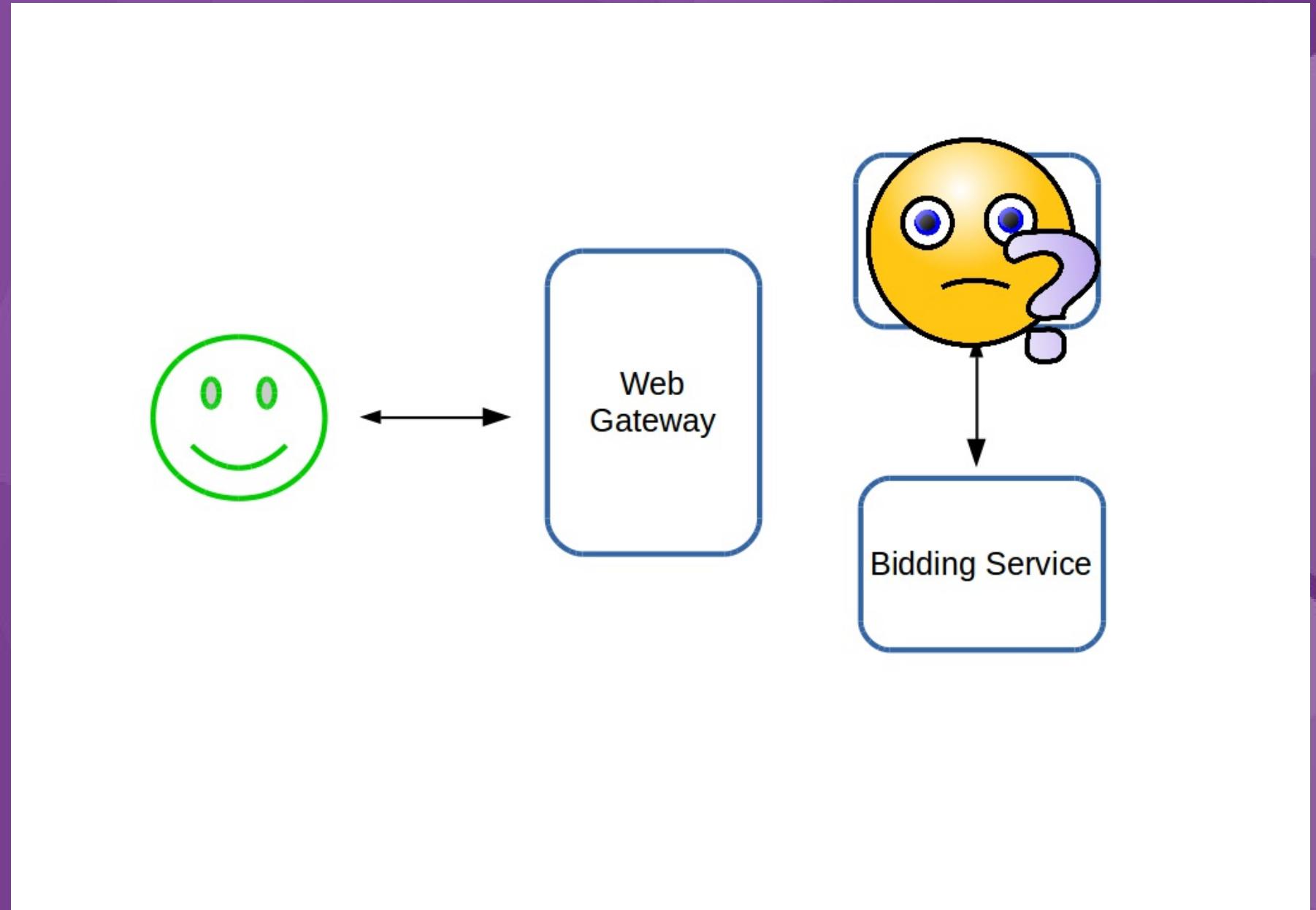
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



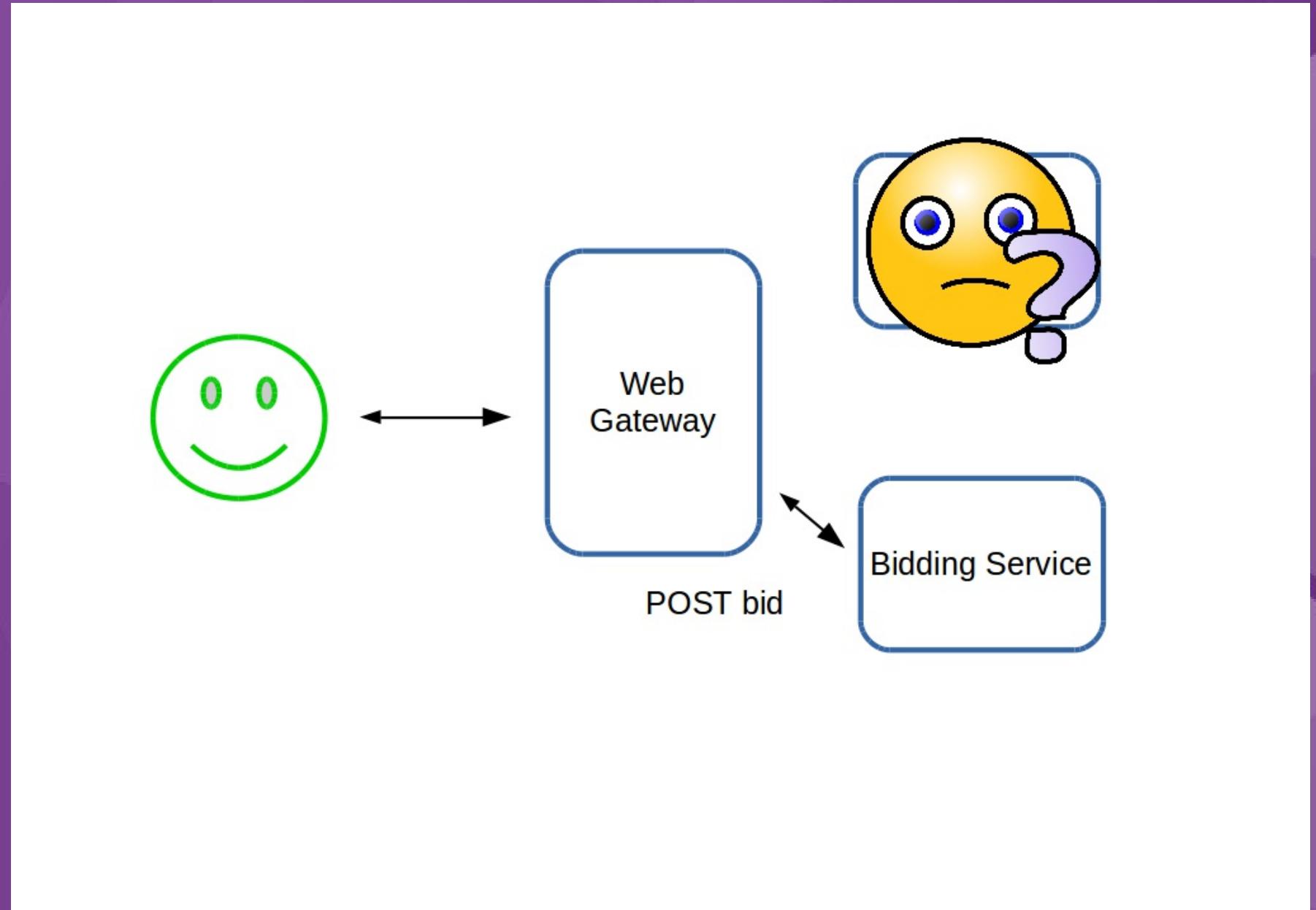
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



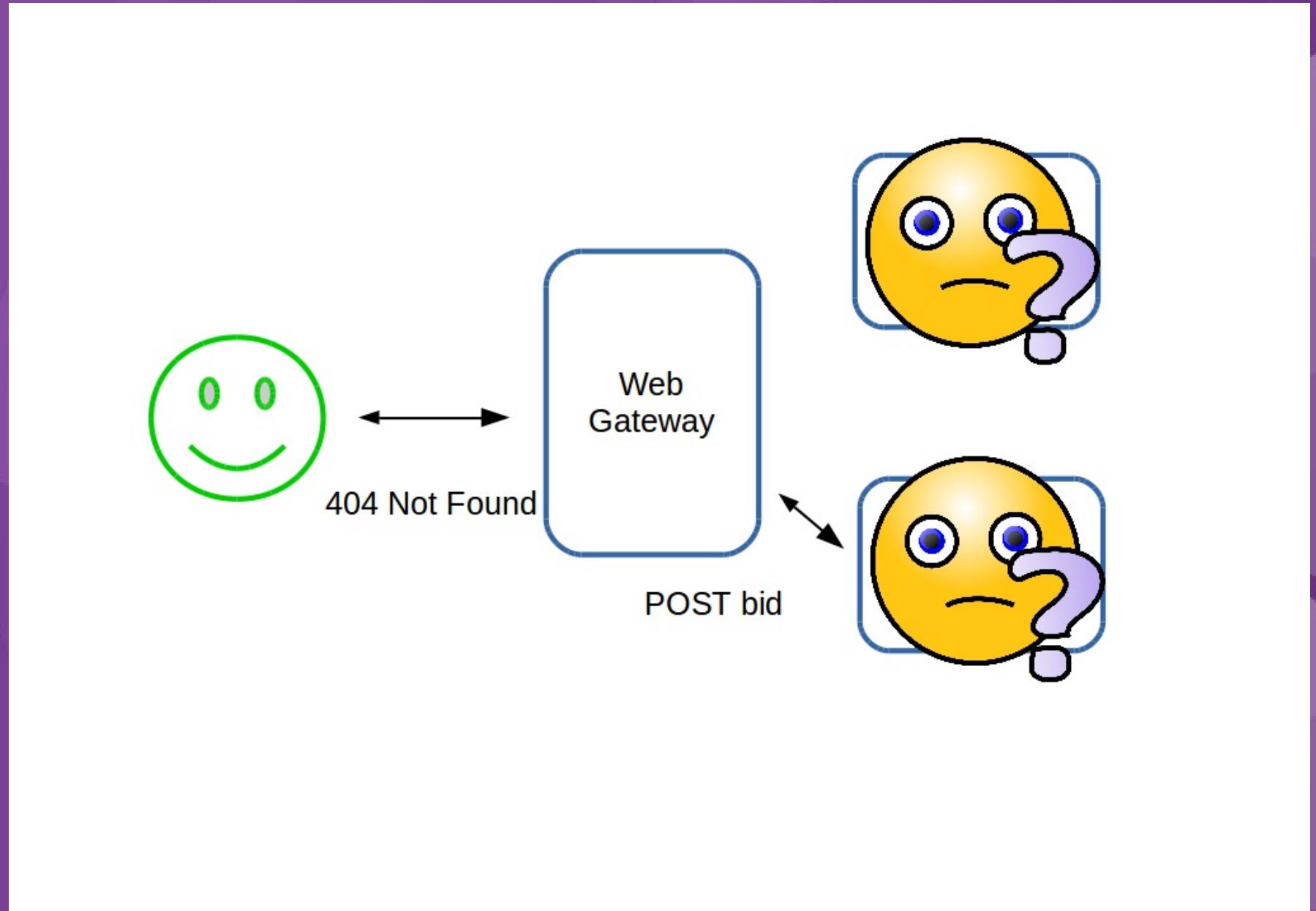
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



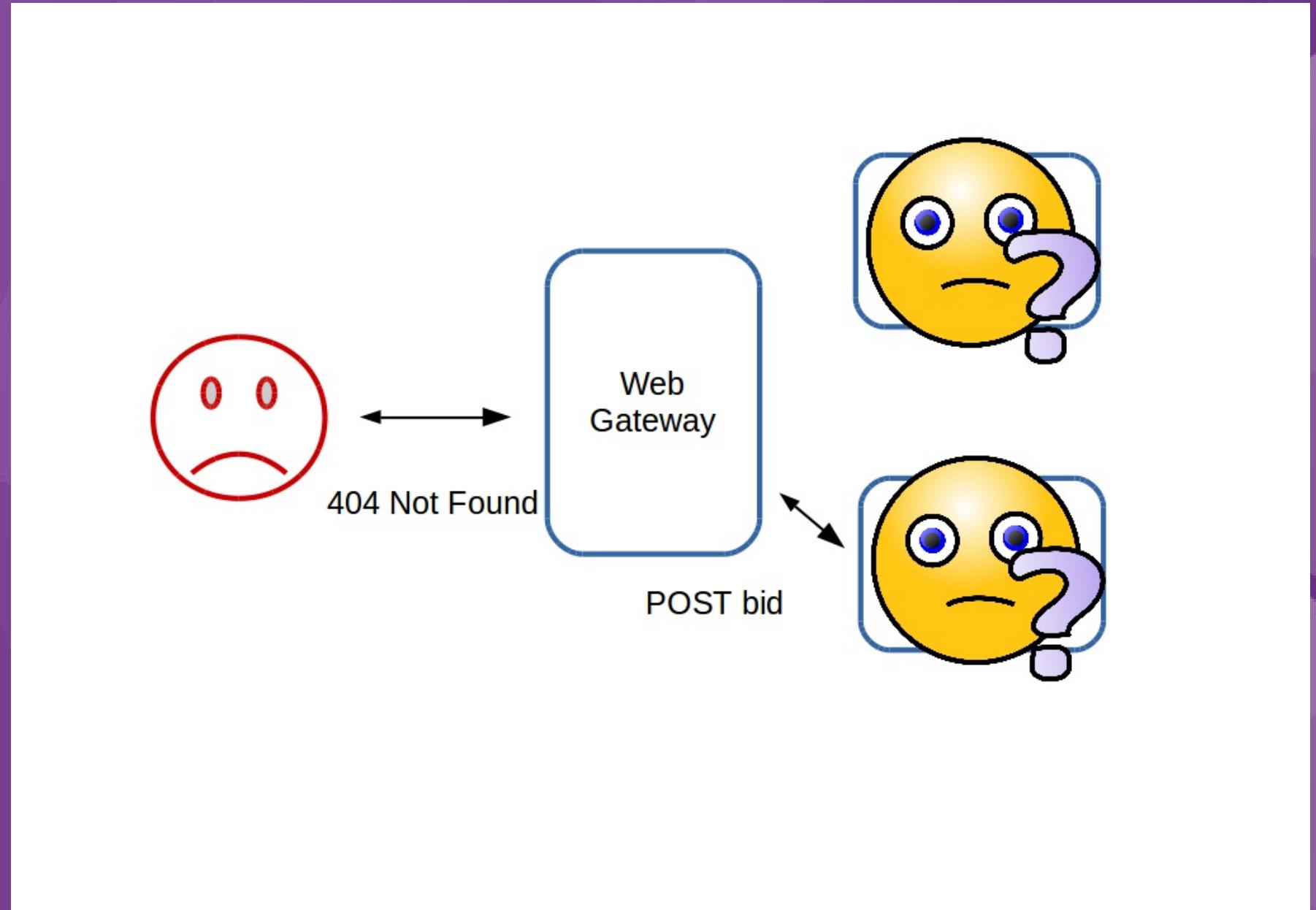
障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない



障害により一貫性が失われる

オークションの開始が入札サービスの障害により伝わらず、ユーザが入札できない

INCONSISTENCY FROM FAILURE

同期通信で「同時」に更新をかけるのは危険

サービス境界を超えるようなトランザクションはできない

INCONSISTENCY FROM FAILURE

- Synchronous "at same time" communication of updates is dangerous

同期通信で「同時」に更新をかけるのは危険

サービス境界を超えるようなトランザクションはできない

INCONSISTENCY FROM FAILURE

- Synchronous "at same time" communication of updates is dangerous
 - Transactions can't span service boundaries

同期通信で「同時」に更新をかけるのは危険

サービス境界を超えるようなトランザクションはできない

PATTERN 3: ASYNCHRONOUS MESSAGING

非同期メッセージング: 全てのシステムが同時に即応性を保つ必要はない

イベントを永続化してメッセージのソースとして使う

PATTERN 3: ASYNCHRONOUS MESSAGING

- Does not require both systems to be responsive

非同期メッセージング: 全てのシステムが同時に即応性を保つ必要はない

イベントを永続化してメッセージのソースとして使う

PATTERN 3: ASYNCHRONOUS MESSAGING

- Does not require both systems to be responsive
- Perfect if you already persist events

非同期メッセージング: 全てのシステムが同時に即応性を保つ必要はない

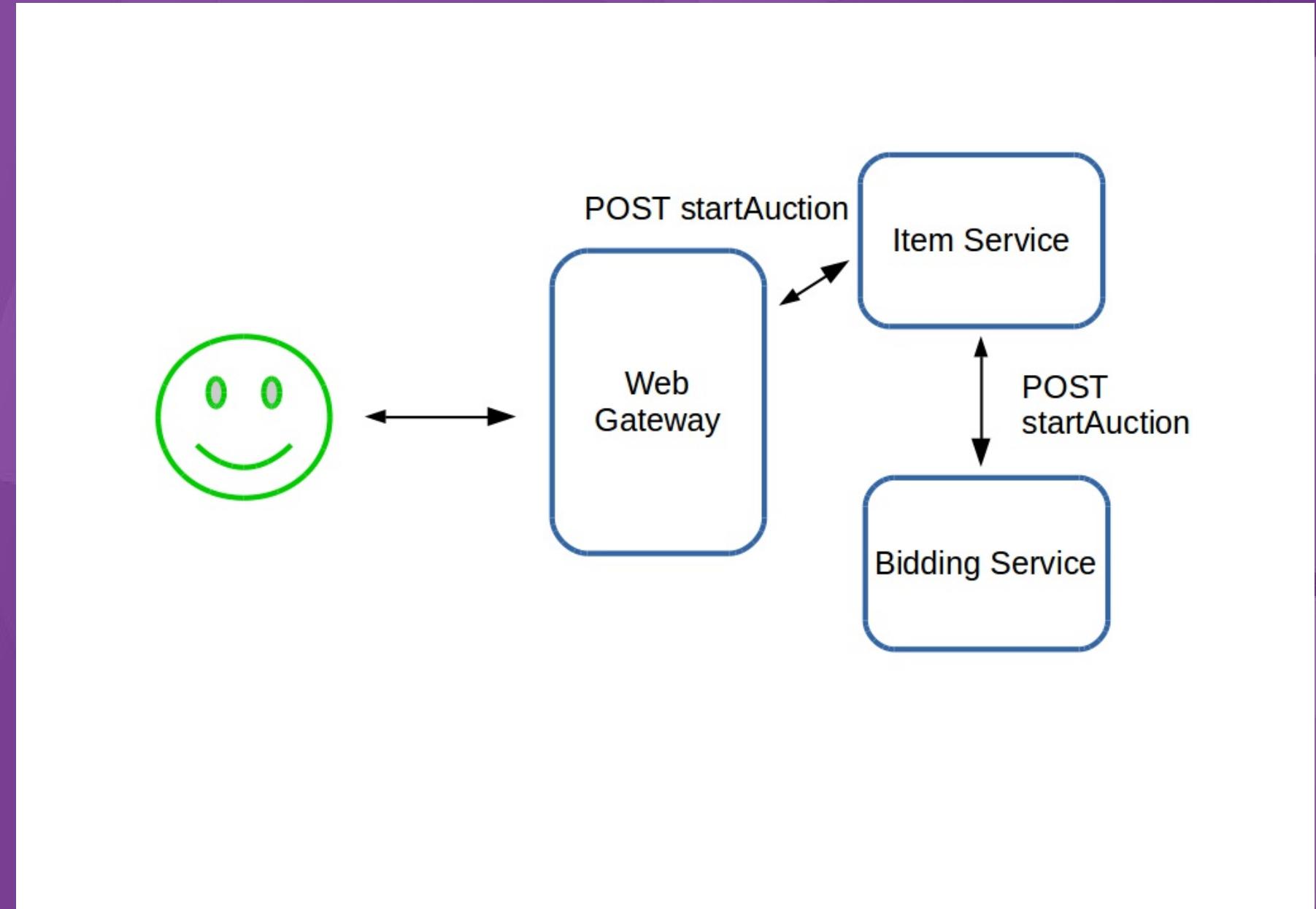
イベントを永続化してメッセージのソースとして使う

PATTERN 3: ASYNCHRONOUS MESSAGING

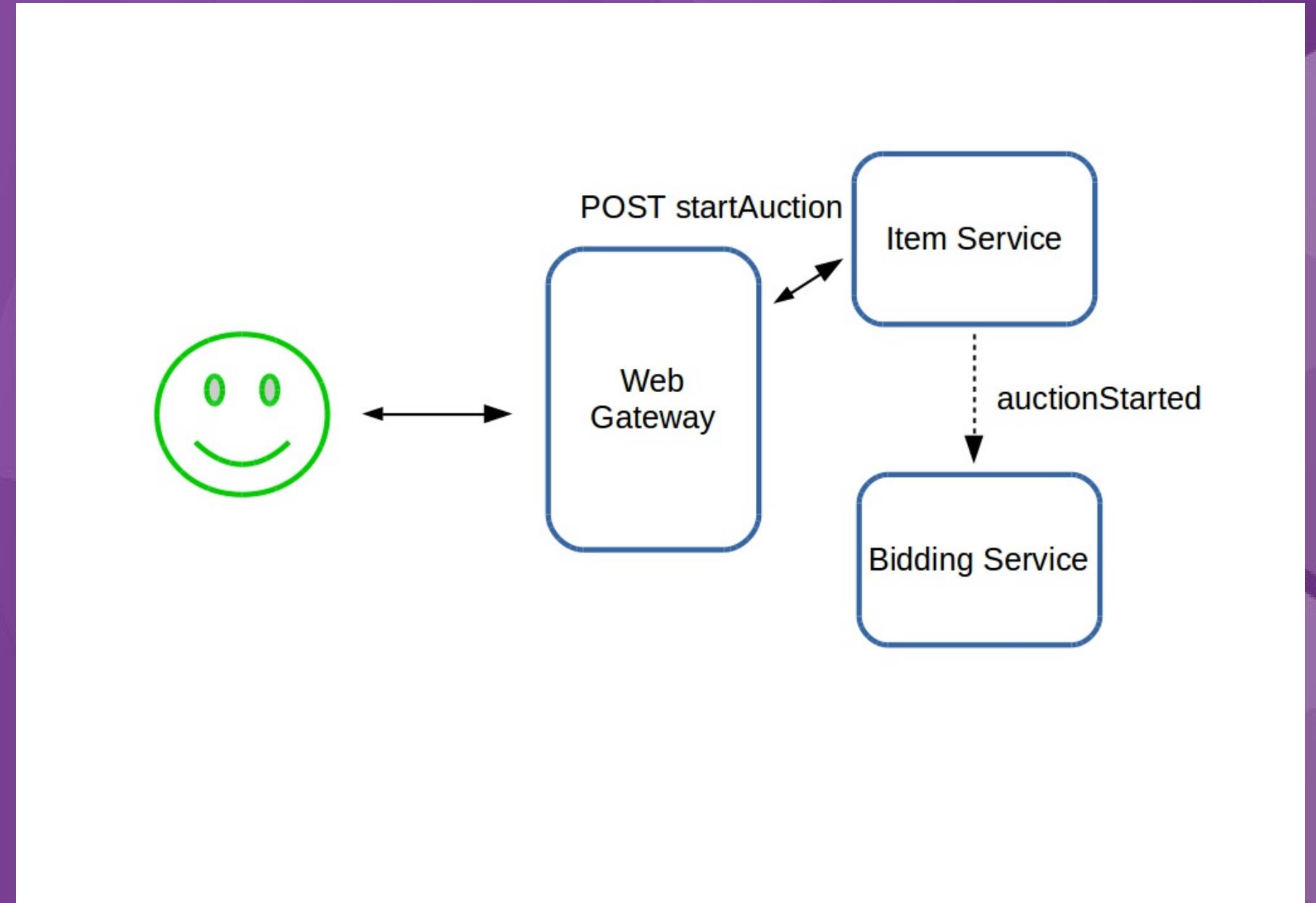
- Does not require both systems to be responsive
- Perfect if you already persist events
- Use persistent events as a source of messages

非同期メッセージング: 全てのシステムが同時に即応性を保つ必要はない

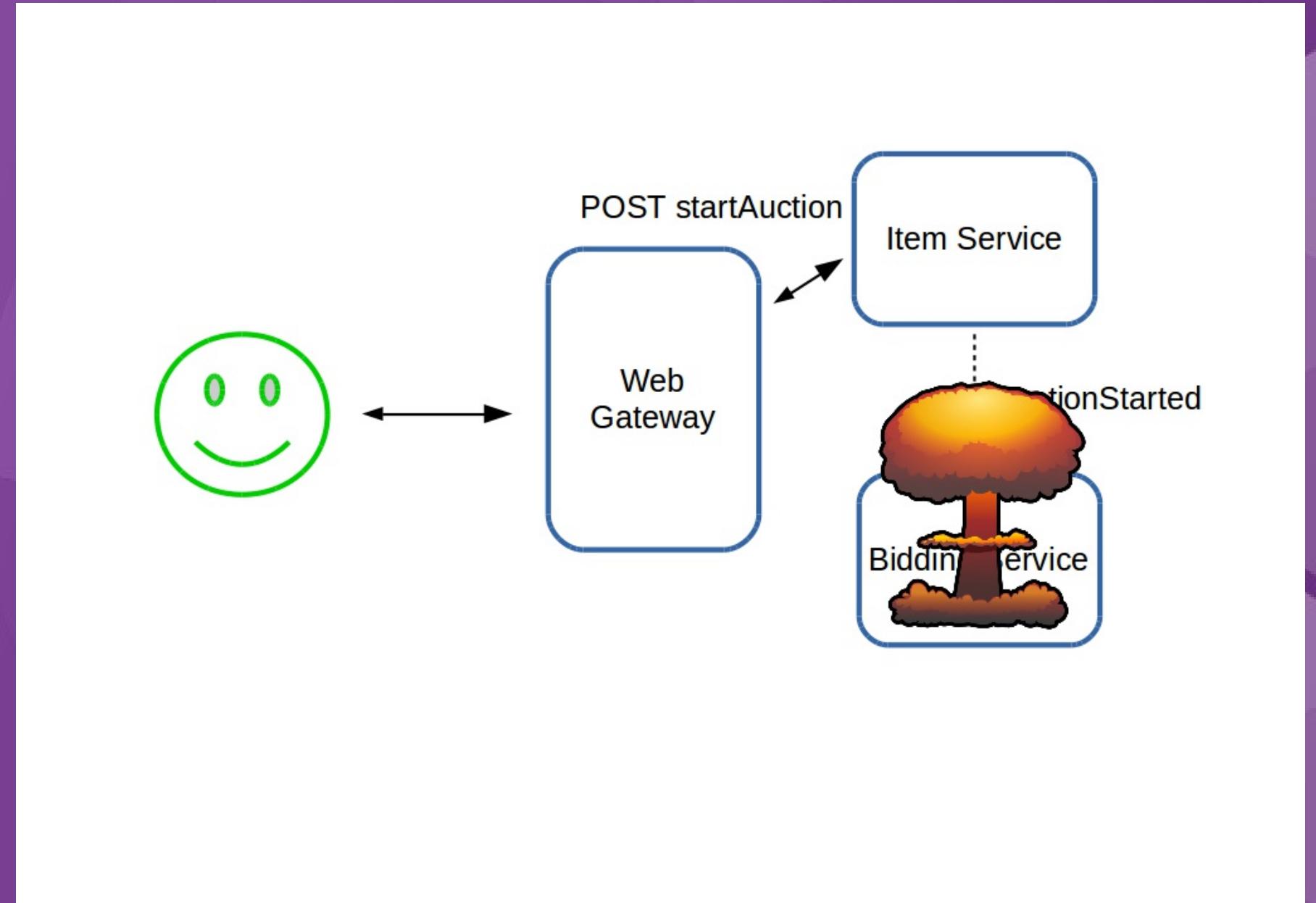
イベントを永続化してメッセージのソースとして使う



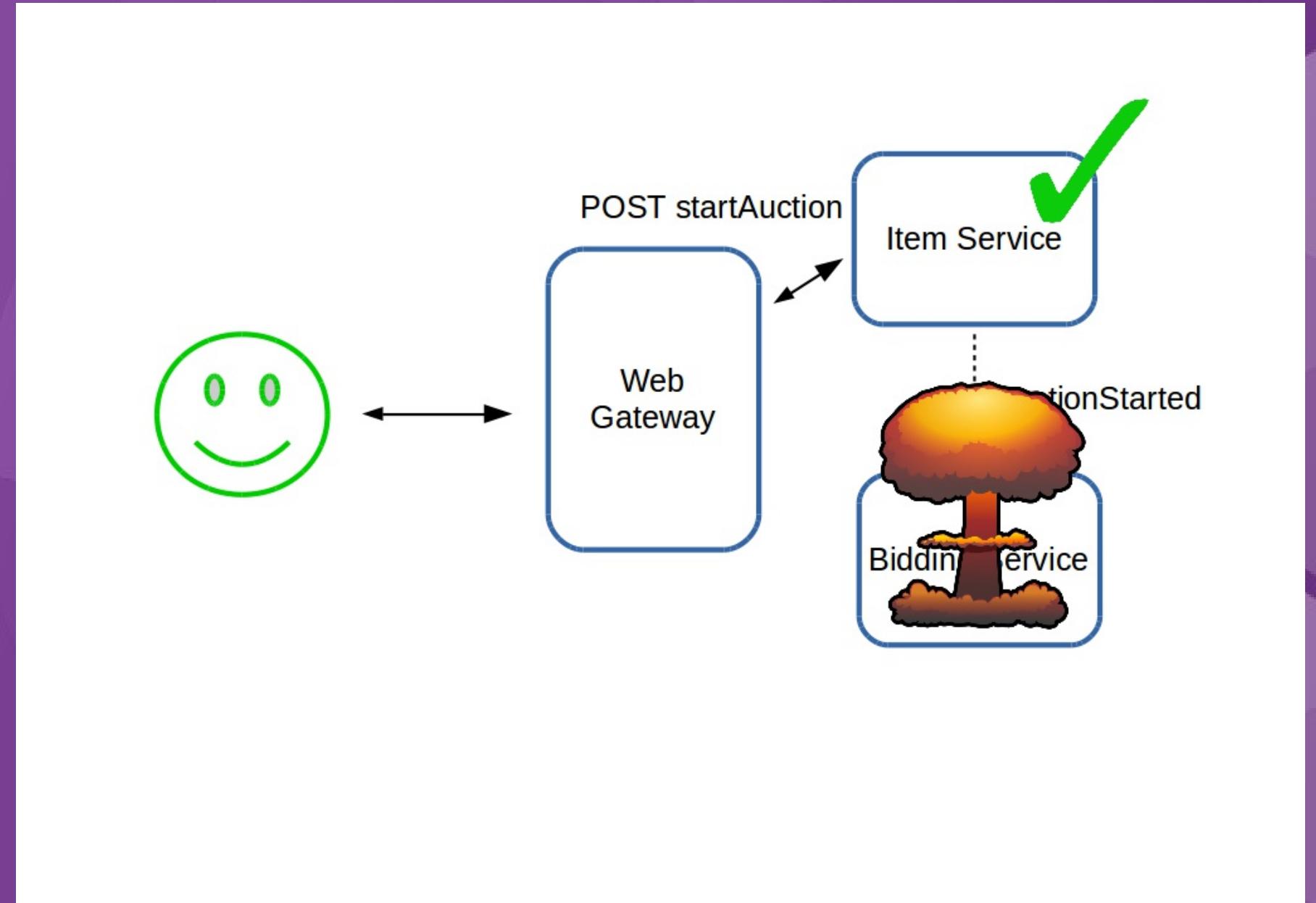
オークション開始のメッセージは、入札サービスに非同期で伝わる
入札サービスが障害を起こしても、復旧すればユーザは入札できる



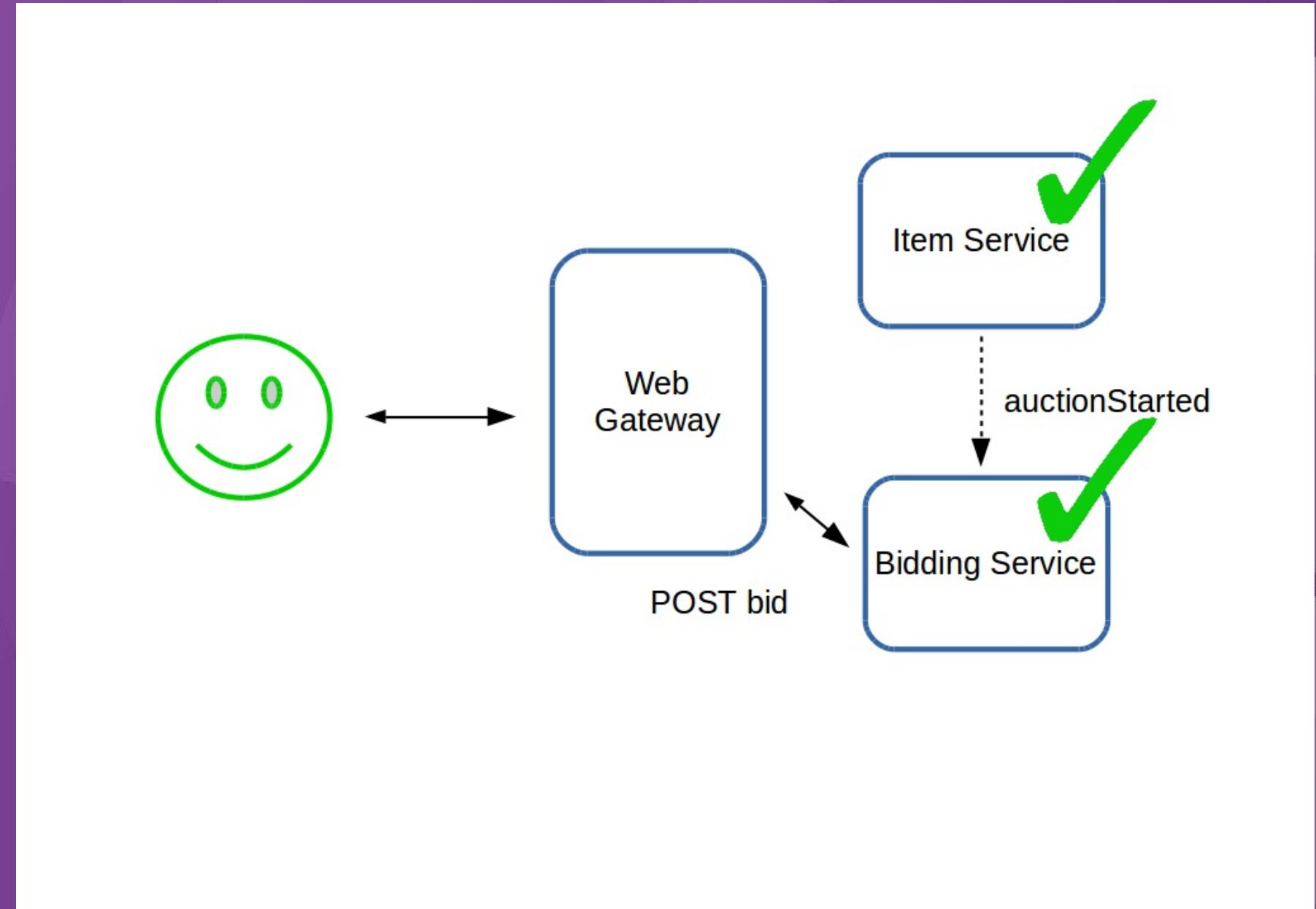
オークション開始のメッセージは、入札サービスに非同期で伝わる
入札サービスが障害を起こしても、復旧すればユーザは入札できる



オークション開始のメッセージは、入札サービスに非同期で伝わる
入札サービスが障害を起こしても、復旧すればユーザは入札できる



オークション開始のメッセージは、入札サービスに非同期で伝わる
入札サービスが障害を起こしても、復旧すればユーザは入札できる



オークション開始のメッセージは、入札サービスに非同期で伝わる
入札サービスが障害を起こしても、復旧すればユーザは入札できる

UNACCEPTABLE DEGRADATION

許容できないデグレード: 入札履歴が0件になるのはともかく

入札価格が0ドルにフォールバックするのはダメ

UNACCEPTABLE DEGRADATION

- Earlier we degraded the item page with empty bid history

許容できないデグレード: 入札履歴が0件になるのはともかく

入札価格が0ドルにフォールバックするのはダメ

UNACCEPTABLE DEGRADATION

- Earlier we degraded the item page with empty bid history
- Price was also \$0

許容できないデグレード: 入札履歴が0件になるのはともかく

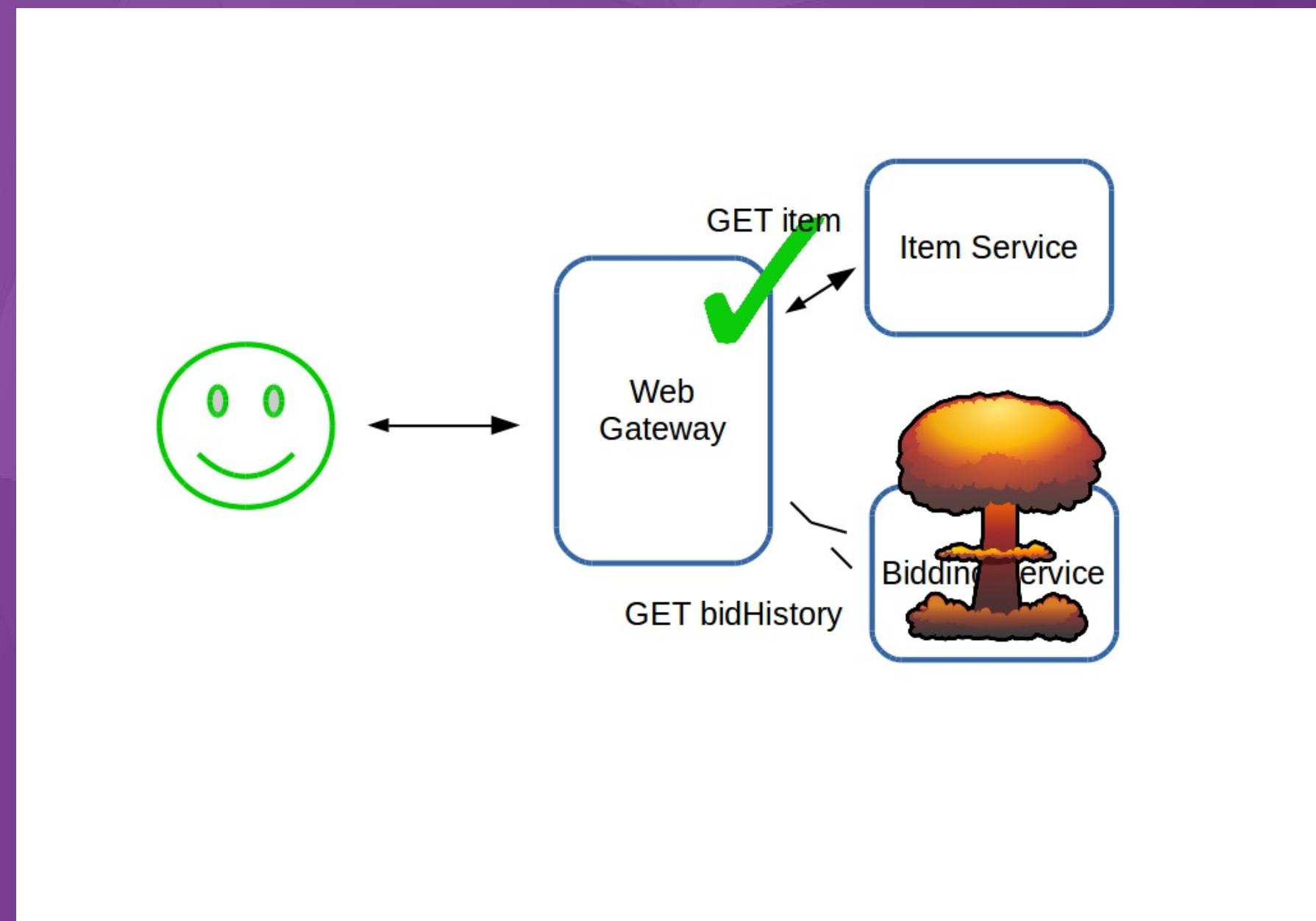
入札価格が0ドルにフォールバックするのはダメ

UNACCEPTABLE DEGRADATION

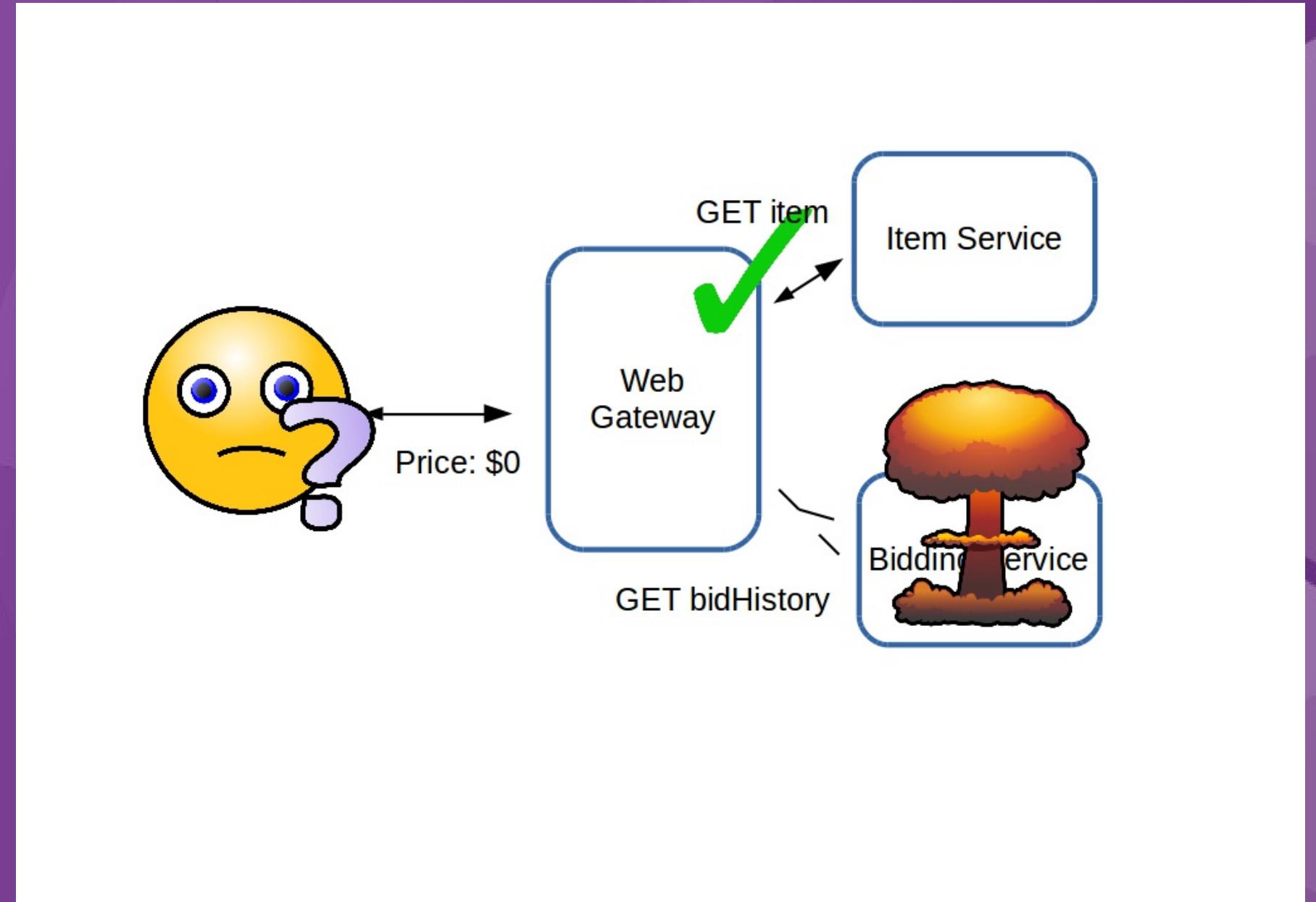
- Earlier we degraded the item page with empty bid history
- Price was also \$0
- Users may tolerate no history, but not wrong price

許容できないデグレード: 入札履歴が0件になるのはともかく

入札価格が0ドルにフォールバックするのはダメ



「えっ、この商品の入札価格 0 ドル？」



「えっ、この商品の入札価格 0 ドル？」

PATTERN 4: DENORMALIZE

非正規化: 他のサービスに重要な情報を送って重複保存する
システムにとって重要なものと、ビジネスにとって重要なものがある

PATTERN 4: DENORMALIZE

- Push important information to other services

非正規化: 他のサービスに重要な情報を送って重複保存する
システムにとって重要なものと、ビジネスにとって重要なものがある

PATTERN 4: DENORMALIZE

- Push important information to other services
 - Important for system functions

非正規化: 他のサービスに重要な情報を送って重複保存する

システムにとって重要なものと、ビジネスにとって重要なものがある

PATTERN 4: DENORMALIZE

- Push important information to other services
 - Important for system functions
 - Important for business functions

非正規化: 他のサービスに重要な情報を送って重複保存する

システムにとって重要なものと、ビジネスにとって重要なものがある

PATTERN 4: DENORMALIZE

- Push important information to other services
 - Important for system functions
 - Important for business functions
- Store duplicated information in those services

非正規化: 他のサービスに重要な情報を送って重複保存する

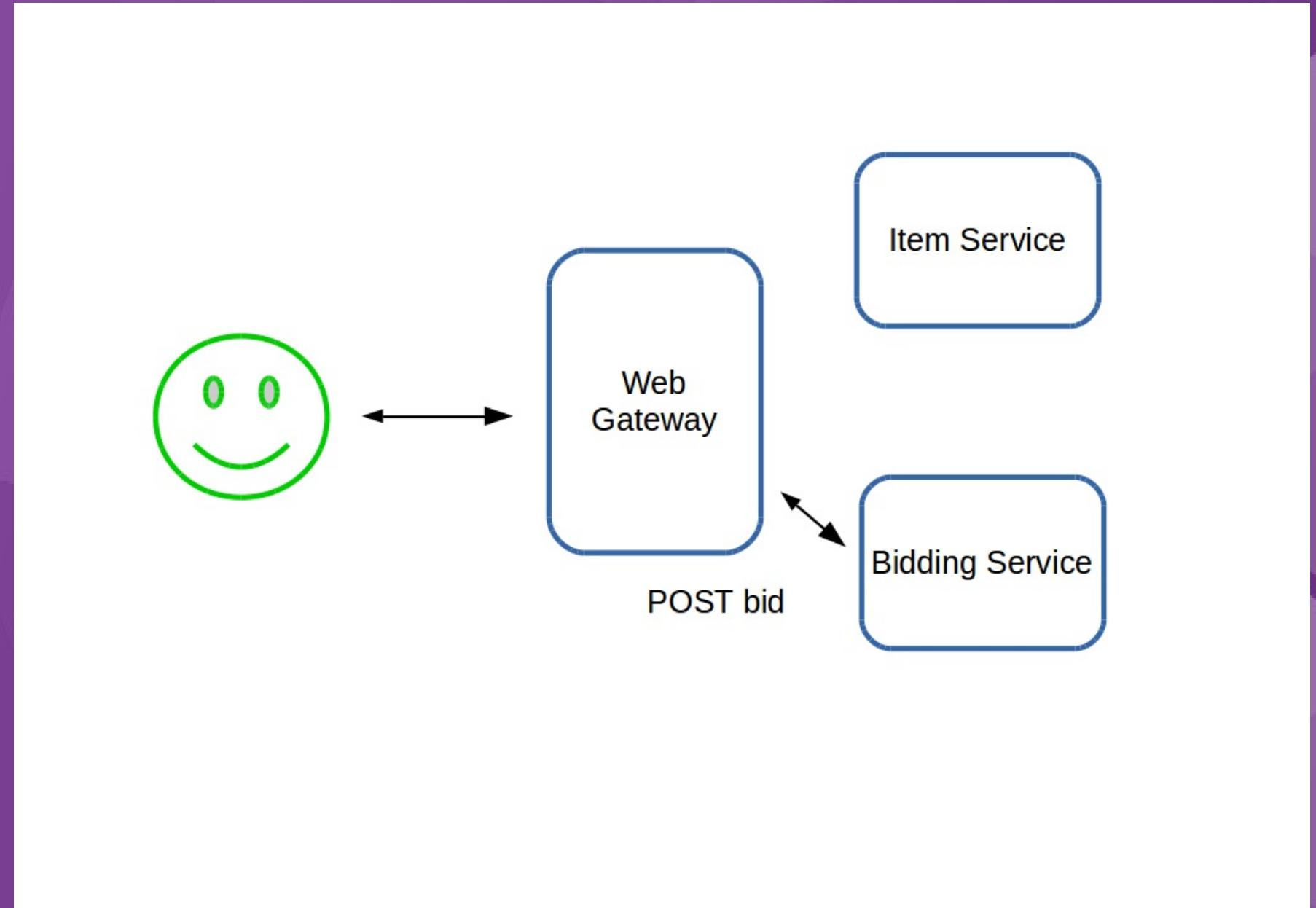
システムにとって重要なものと、ビジネスにとって重要なものがある

PATTERN 4: DENORMALIZE

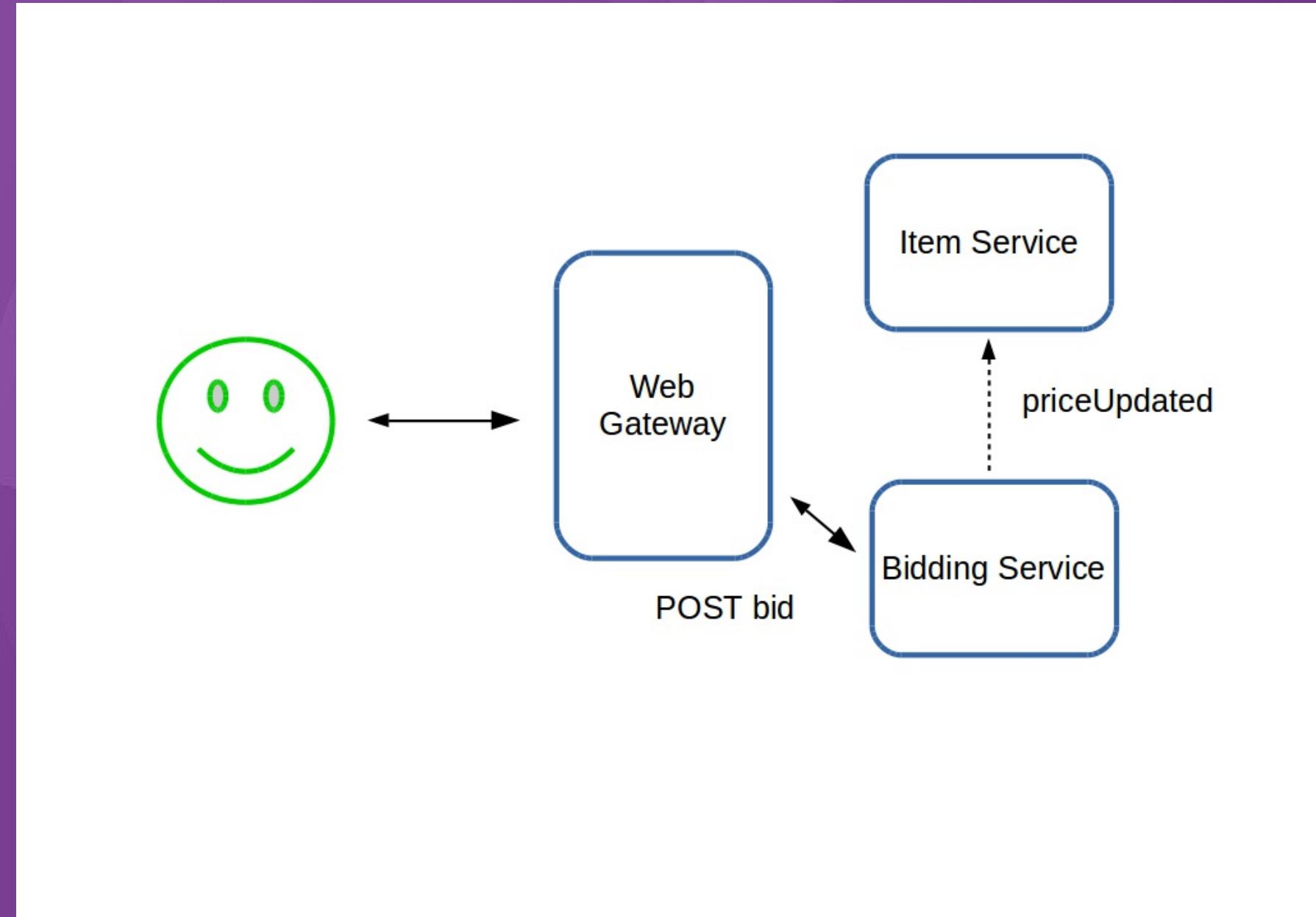
- Push important information to other services
 - Important for system functions
 - Important for business functions
- Store duplicated information in those services
 - AKA denormalization

非正規化: 他のサービスに重要な情報を送って重複保存する

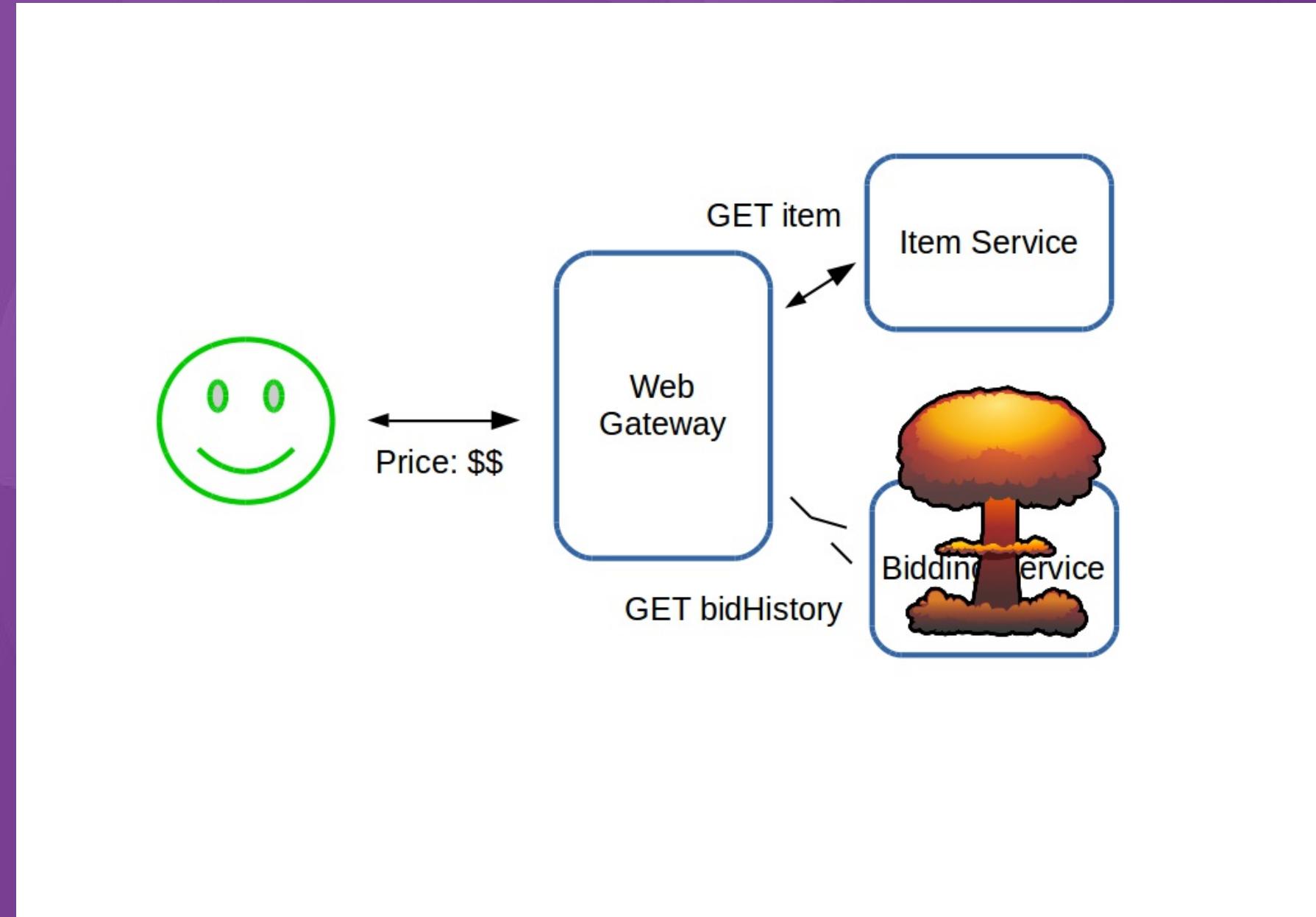
システムにとって重要なものと、ビジネスにとって重要なものがある



入札価格が更新されたことを商品サービスにも伝えておく
入札サービスが障害を起こしても、商品サービスから入札価格を取得できる



入札価格が更新されたことを商品サービスにも伝えておく
入札サービスが障害を起こしても、商品サービスから入札価格を取得できる



入札価格が更新されたことを商品サービスにも伝えておく
入札サービスが障害を起こしても、商品サービスから入札価格を取得できる

SUMMARY

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows
- Failure and inconsistency must be managed

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows
- Failure and inconsistency must be managed
- Use:

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows
- Failure and inconsistency must be managed
- Use:
 - Circuit breakers

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows
- Failure and inconsistency must be managed
- Use:
 - Circuit breakers
 - Failure degradation

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows
- Failure and inconsistency must be managed
- Use:
 - Circuit breakers
 - Failure degradation
 - Asynchronous messaging

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が
必要

障害と一貫性の崩れは管理する必要がある

SUMMARY

- Monolith to microservices requires rearchitecting data flows
- Failure and inconsistency must be managed
- Use:
 - Circuit breakers
 - Failure degradation
 - Asynchronous messaging
 - Denormalization

モノリスをマイクロサービスに置き換えるにはデータフローの再設計が必要

障害と一貫性の崩れは管理する必要がある

NEXT STEPS

<http://lagomframework.com>

<https://github.com/jroper/microservices-architecture-scala-jp>