Case Study B:
"Concurrent Updates in WPF"

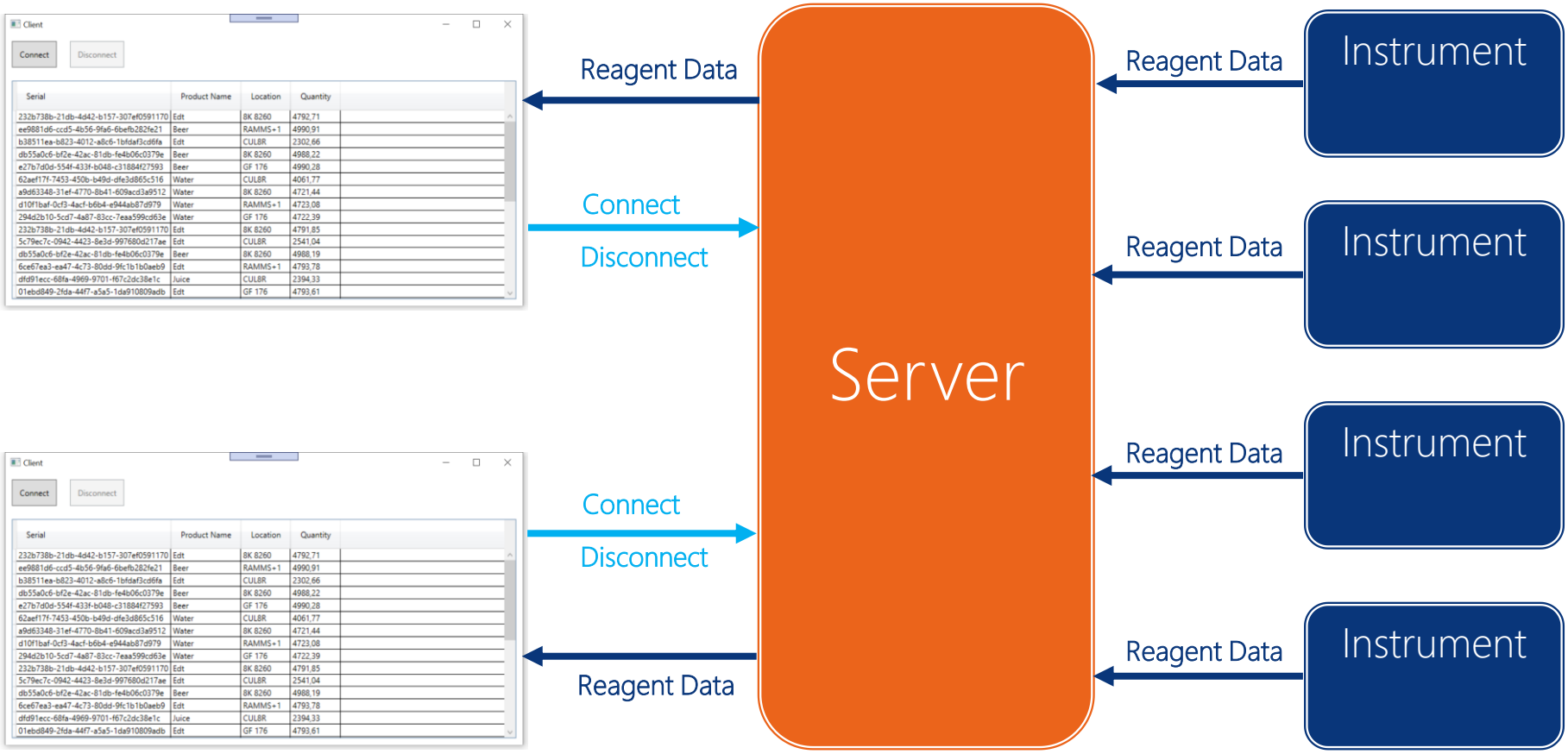
TEKNOLOGISK
INSTITUT

Agenda

- Case Study Scenario
- V2: A Step in the Right Direction
- V3: Improving the Client Display
- V4: To Lock or not to Lock?
- V5: Are We Happy (Enough)?

TEKNOLOGISK INSTITUT

Scenario Overview



TEKNOLOGISK INSTITUT

Scenario Description

- Server hosts two self-hosted WCF services
  - Instrument Service
    - Instruments send Reagent Data
    - One-way using HTTP
  - Client Service
    - Clients (dis)connect
    - Receives Reagent Data live as it arrives from instruments
      - callback to connected clients
    - Two-way using TCP

Discussion

- Is the system thread-safe?
  - Needs careful scrutiny to decide
- Are the instruments thread-safe?
- Are the clients thread-safe?
- Is the Server thread-safe?

Client App

- Yes, it is thread-safe! But very subtle indeed…!
- One very important aspect to consider
  - WCF Synchronization Context
- `ServiceBehavior` + `CallbackBehaviour`
  - `UseSynchronizationContext` is true by default

Service App

- Another very important aspect to consider
  - WCF Instancing Mode vs. Concurrency Mode

- `ServiceBehavior` + `CallbackBehaviour`
  - `InstanceContextMode.Single` => `ConcurrencyMode.Single`

- So, is the Service also thread-safe?? ☺
  - Very close, but…

Service App Problems

- …Not quite thread-safe, but almost! ☺
  - The list of connected clients is accessed from both sides!
  - Is it a good solution to lock list of connected clients?

- Even worse: Can it deadlock? Why (not)?

TEKNOLOGISK INSTITUT

Agenda

- Case Study Scenario
- **V2: A Step in the Right Direction**
- V3: Improving the Client Display
- V4: To Lock or not to Lock?
- V5: Are We Happy (Enough)?

TEKNOLOGISK INSTITUT

A Step in the Right Direction: Go Asynchronous!

- Make all WCF client proxies call the services asynchronously!
- Use **`async`** + **`await`** on caller side

```
interface IClientAsyncContract
{
    ...
    void Connect();
    Task ConnectAsync();
    ...
}
```

- Make all contract methods asynchronous
  - In interfaces and implementation

Can Services be Asynchronous too?

- WCF is quite clever with respect to asynchrony
  - Asynchrony of client side is **independent (*)** of
  - Asynchrony of service implementation
  - Can support both!
- Might as well change internally in Server also…!
  - A ”pure” asynchronous flow emerges
- Is everything great now?

WTF? UI Buttons are Not Updating…?!

- Another extremely subtlety rears its ugly face!
- In MVVM the RelayCommand (a.k.a. DelegateCommand) uses the WPF CommandManager to reevaluate command enabledness
  - CommandManager.RequerySuggested
- Threading and asynchronous occasionally "confuses" the CommandManger in WPF
- Solution is to manually instruct CommandManager to test

```
// Forcing the CommandManager to raise the RequerySuggested event
CommandManager.InvalidateRequerySuggested();
```

TEKNOLOGISK INSTITUT

Agenda

- Case Study Scenario
- V2: A Step in the Right Direction
- **V3: Improving the Client Display**
- V4: To Lock or not to Lock?
- V5: Are We Happy (Enough)?

Doing a Better Client View

- There are many benefits of the client synchronization context being the **`DispatcherSynchronizationContext`**!

- Can now easily turn display client more intelligent
  - No locking necessary

TEKNOLOGISK INSTITUT

Agenda

- Case Study Scenario
- V2: A Step in the Right Direction
- V3: Improving the Client Display
- **V4: To Lock or not to Lock?**
- V5: Are We Happy (Enough)?

TEKNOLOGISK INSTITUT

Back to Thread-Safety of Server

- Several possible solutions exist
  1. ~~Locking the entire list while calling out?~~
  2. ~~Locking the entire list whenever Reagent Data arrives from Instrument?~~
  3. ~~Creating a Server-wide common SynchronizationContext~~
     i. ~~Used for calls in both services by "installing" it into WCF~~
  4. Copying the list before calling out
     i. Living with **`ObjectDisposedException`** if client disconnects in the meantime
     ii. Introduce some additional parallelism
  5. Locking the list only when starting the callback tasks
     i. Unlock before awaiting each call completion
     ii. Introduce some additional parallelism
  6. …

- What are the pros and cons of each solution?

TEKNOLOGISK INSTITUT

Some Notes on the Client

- All solutions are crucially (negatively and positively) impacted by the client synchronization context being the **`DispatcherSynchronizationContext`**!

- Will probably have to live with **`ObjectDisposedException`** in such a system no matter what…
  - If you let clients disconnect while a Reagent Data update from Instrument is being processed

TEKNOLOGISK INSTITUT

Agenda

- Case Study Scenario
- V2: A Step in the Right Direction
- V3: Improving the Client Display
- V4: To Lock or not to Lock?
- **V5: Are We Happy (Enough)?**

TEKNOLOGISK INSTITUT

Further Considerations

- What if clients need to communicate, e.g. Notes?
- What if some client is slooow in processsing updates?
- What about having the Server hold the current state and synchronize it to clients upon connect?
- What about disconnections and missing updates?
- What about increasing throughput?
- What about reentrancy?
- What about consistent ordering with more concurrency?
- What about Server crashes?
- What about…? ☺

"Good Enough is Good Enough" ☺

- Each step
  - increases performance
  - increases complexity
  - Increases development time
  - …
- Might even consider a persistent event store

TEKNOLOGISK INSTITUT

Summary

- Case Study Scenario
- V2: A Step in the Right Direction
- V3: Improving the Client Display
- V4: To Lock or not to Lock?
- V5: Are We Happy (Enough)?


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Ringgårdsvej 4A
8270 Højbjerg
Denmark