1.Possible Root Causes

Code-level causes

- **Event not firing:** TelemetryReceived or DeviceDisconnected events in DeviceSimulator may not trigger if exceptions are thrown in Tick() or timer not started.
- **Dispatcher issue:** UI updates via App.Current.Dispatcher.Invoke() may silently fail if called from a disposed context.
- **Property change not reflected:** Missing or misfired INotifyPropertyChanged in DeviceModel may cause UI binding not to refresh IsOnline.
- **Timer conflicts:** _timer interval too short or long (default 2000 ms) may cause overlapping Elapsed events.
- **Race condition:** Multiple threads updating the same ObservableCollection (Logs or Devices) could occasionally throw or skip updates.

Network or simulation causes

- **Random disconnect simulation:** if (_rand.NextDouble() < 0.10) intentionally sets devices offline (10% chance) — this may appear as "intermittent failure."
- **Device not returning online:** Once marked IsOnline = false, no reconnect logic exists in DeviceSimulator, so devices remain offline.

UI/Binding causes

- **DataContext mismatch:** If MainViewModel not properly bound or replaced in MainWindow.xaml, updates won't appear.
- **ObservableCollection not notifying correctly:** Adding logs or device changes outside UI thread may cause "CollectionModified" exceptions or missing refreshes.
- **UI thread busy:** Continuous logging or frequent updates may block UI rendering.

2.Debugging Plan & Tools

Layer	Method	Tool / Technique
Code (Logic)	Add structured logs inside DeviceSimulator.Tick() to trace random disconnects and telemetry events.	System.Diagnostics.Debug.WriteLine() or Serilog
Network (Simulation)	Force consistent simulation values (disable randomness temporarily).	Comment out _rand.NextDouble() condition to verify stability.
UI (Binding)	Use Visual Studio "Live Visual Tree" and "Live Property Explorer" to confirm IsOnline binding updates.	Visual Studio Debug Tools
Events	Set breakpoints on event invocations (TelemetryReceived, DeviceDisconnected).	Breakpoint + "When Hit" message
Threading	Use Dispatcher.CheckAccess() before UI calls to confirm thread context.	Breakpoint inspection
Logging Panel	Verify that Logs.Add(...) is always called via UI thread (Dispatcher.Invoke).	Console/Output monitoring

Sample Debug Logging Code

```
private void Tick(object? sender, ElapsedEventArgs e)
{
    foreach (var d in _devices)
    {
        try
        {
            System.Diagnostics.Debug.WriteLine($"Tick: {d.Name}
Online={d.IsOnline}");
            if (_rand.NextDouble() < 0.10)
            {
                d.IsOnline = false;
                DeviceDisconnected?.Invoke(this, d);
                continue;
            }
```

```
            if (d.IsOnline)
            {
                d.LastUpdated = DateTime.Now;
                double temp = 20 + _rand.NextDouble() * 10;
                double hum = 40 + _rand.NextDouble() * 20;
                TelemetryReceived?.Invoke(this, new
DeviceTelemetryEventArgs(d, temp, hum));
            }
        }
        catch (Exception ex)
        {
            ErrorOccurred?.Invoke(this, $"Error: {ex.Message}");
        }
    }
}
```