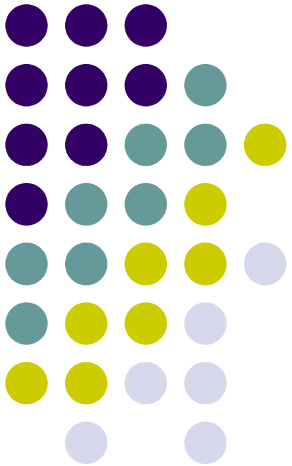
XML on Large Power Transformers and Industrial Batteries

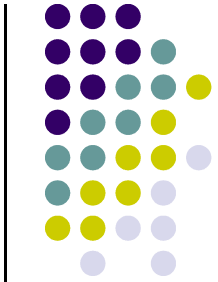
Patrick Cauldwell

Architect

Serveron Corporation

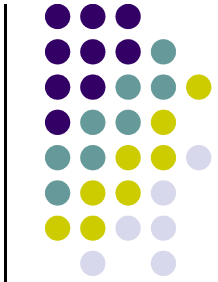


Our Business



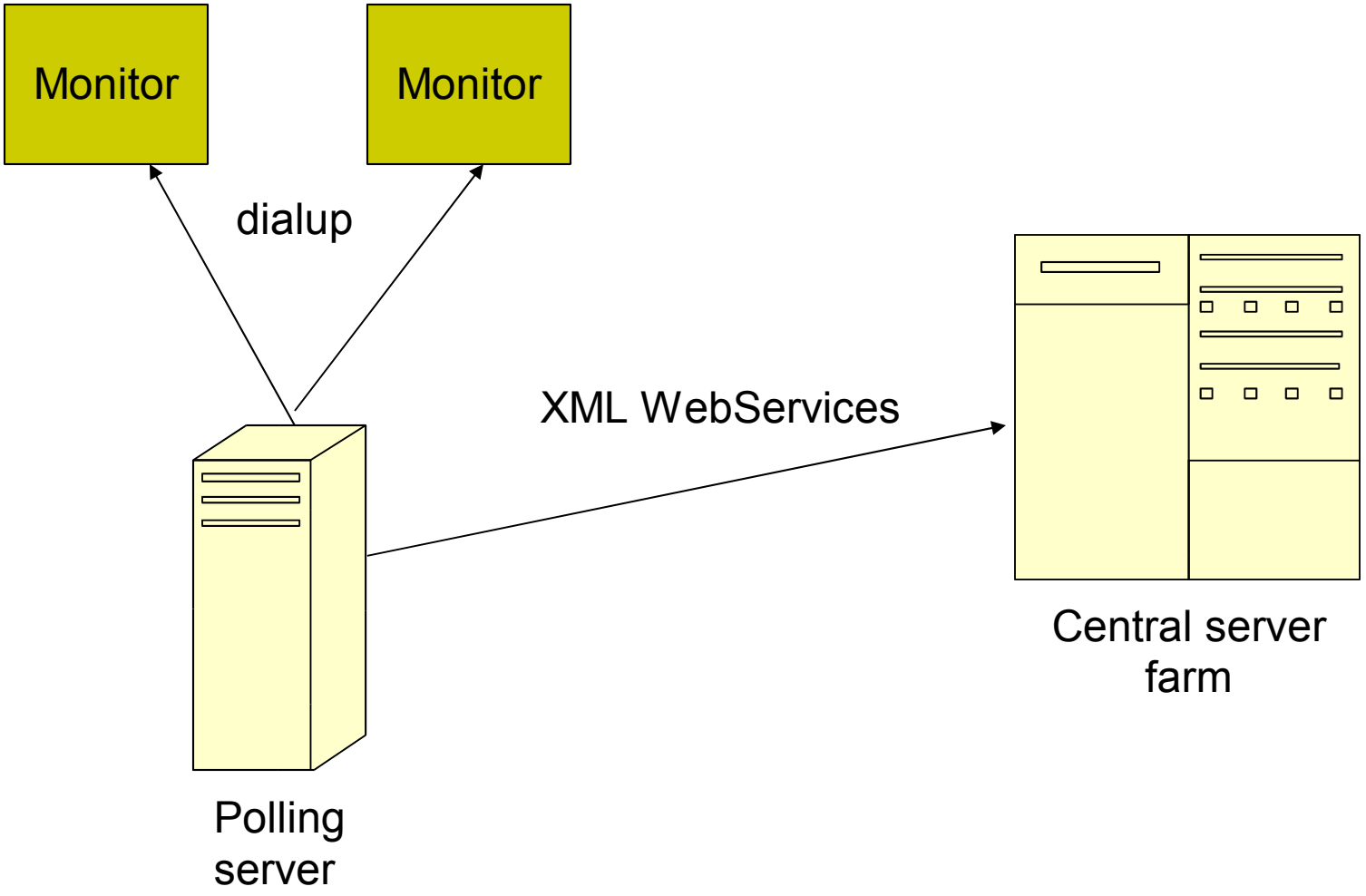
- Monitoring devices
  - Transformer monitors (TrueGas)
  - Battery monitors (BCM)
- Monitoring service
  - Web-based monitoring application
    - Moving to WinForms and WS
  - email or phone notifications

Our current solution

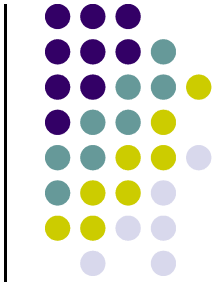


- Data collected from monitors in the field via dialup
- Data sent to central location via WS
- Customers access the central web site
- Service personnel alerted via email/pagers/ SMS

Our current solution

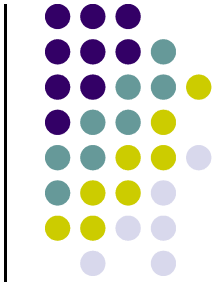


The issues



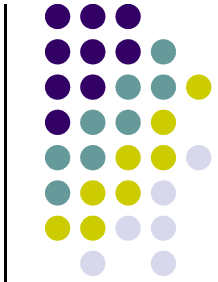
- Doesn’t scale
  - Eventually there are not enough phone lines
- Monitors use proprietary communications formats
  - Too much custom code
- Web interface too limited / not interactive enough
  - Data visualization is UI intensive

The industry standards



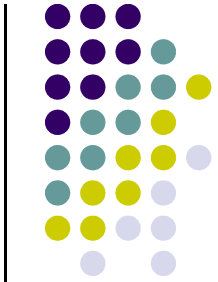
- ModBus
- DNP3
- UCA
- JINI???
- XML Web Services

The solution



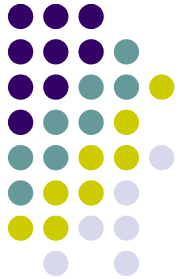
- Monitors push data via WS
  - Data formatted as XML
  - Not limited by phone lines
  - Platform independent
  - Takes advantage of standards based services (GXA)

The constraints



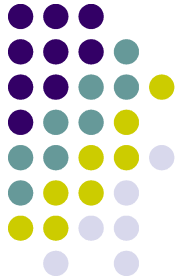
- Limited resources on monitor
  - Limited memory
  - Difficult to maintain state between runs
  - Limited processing power
- Monitors may not be reachable
  - Firewalls are everywhere
  - Dialup is outbound from monitor

Typical XML Web Services models



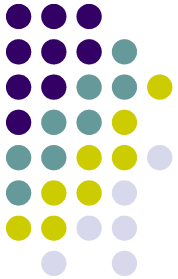
- Server to server interchange (B2B)
  - Both client and server are resource rich
  - Both client and server are “reachable”
  - “Chunky” to save on network overhead
- Smart client
  - Client may be resource limited
  - BUT
  - Client typically requests data from server

Our model



- Monitors are assumed to be unreachable
  - “clients” always push data
  - All SOAP requests originate at the monitor
- Requests are chatty to save memory on the monitor
- Monitor doesn’t maintain any state about what data it sends

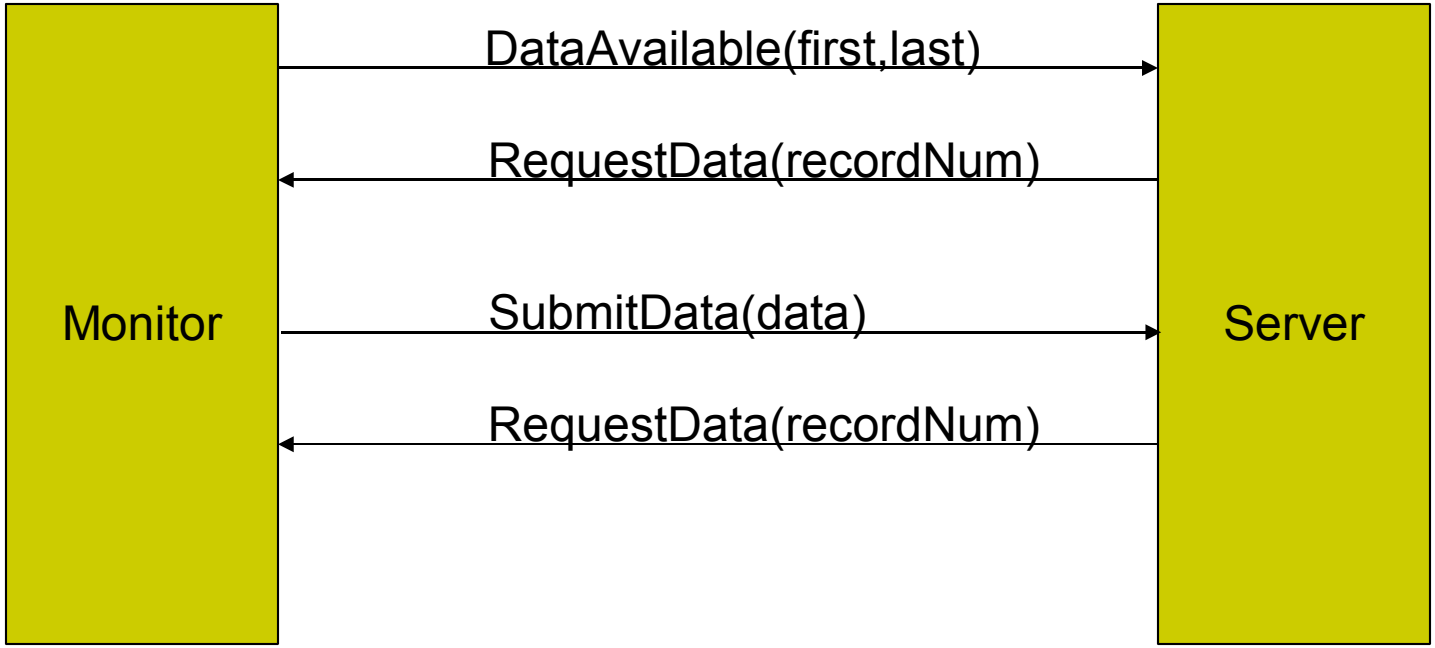
The result



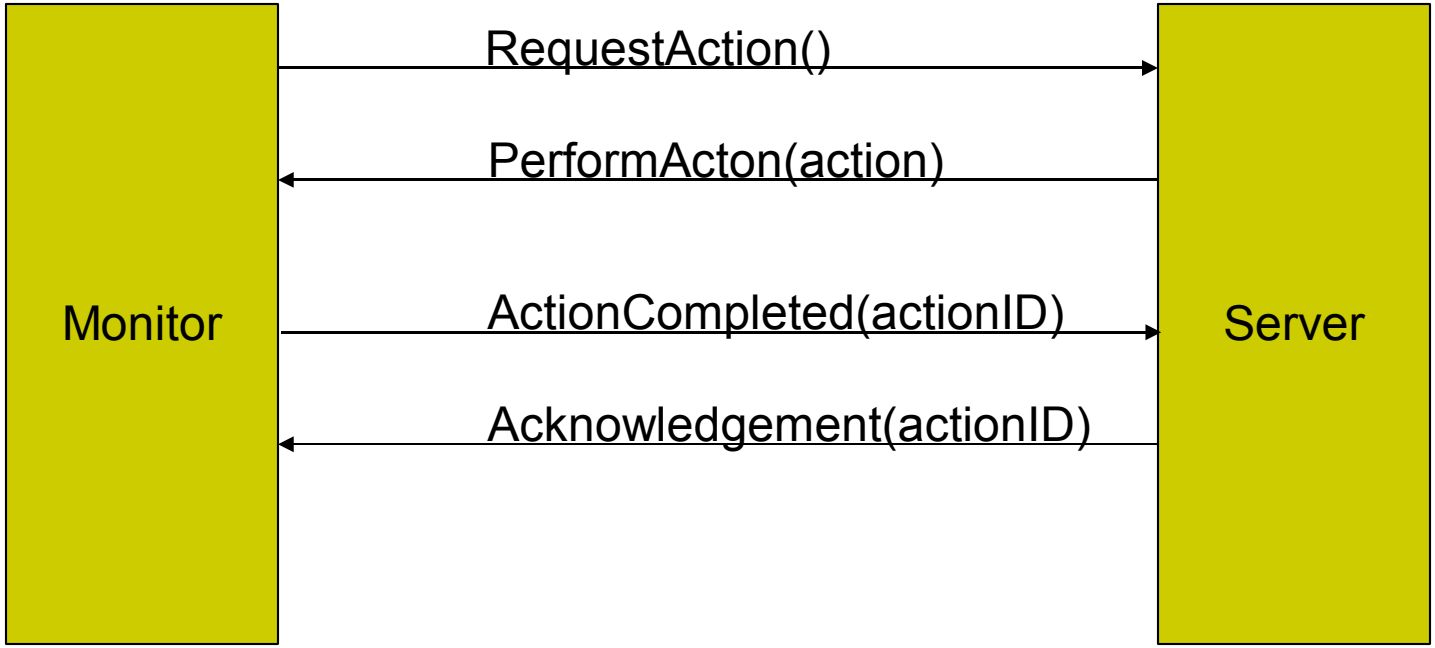
- Our API is “backward”
  - Rather than Data GetData(latestData)
  - We have
  - RequestForData DataAvailable(latestData)

  - Rather than Result SetConfiguration(config)
  - We have
  - Config ConfigRequest()

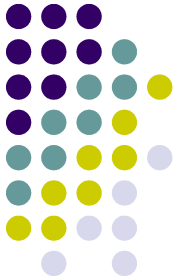
Interaction



More interaction

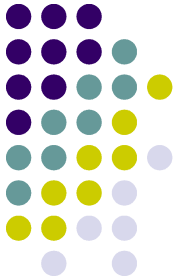


Monitor implementation



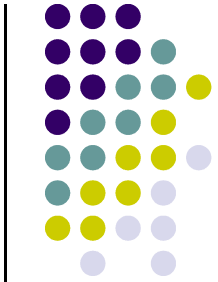
- No state maintained about data sent
  - Supports multiple servers
- Chatty protocol saves on heap space
- Limited parsing and SOAP tools available for embedded systems
  - Expat
  - EasySOAP
  - Most other tools are too big (Xerces, ApacheSOAP)
  - Widest breadth of tools available in Java and .NET
  - Next to no support for GXA

Data structure



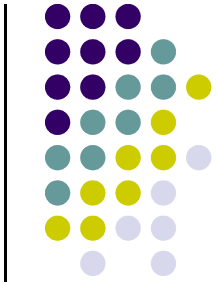
- Generic “sensor” model
- Similar in intent to OpenGIS SensorML
- WSDL is left extensible
  - WSDL doesn’t need to be “discoverable” so the schema need not be definitive

Server implementation



- All .NET
- No session state
- Good tools support for XML Web Services
- Emerging support for GXA (WSE)
- Default for SOAP model is document/literal

Conclusion



- Constraints of the embedded system mean a very different model than server to server
- Data flows from “client” to “server”, not vice versa
- Little tools support on the client side
- API must be designed to be a common ground between the two platforms