Modelo para um Sistema de Gerenciamento de alertas hidrológicos em uma arquitetura baseada em serviços

Armando Câmara Junior [1] José Luiz Moreira [1]
Centro de Previsão de Tempo e Estudos Climáticos, (CPTEC)
Instituto Nacional de Pesquisas Espaciais (INPE)
Cachoeira Paulista – SP – Brasil

armando.camara@cptec.inpe.br , jose.luiz@sir.inpe.br

Resumo. *Este artigo descreve um sistema de gerenciamento de alertas hidrológicos em uma arquitetura baseada em serviços. Esta estrutura é formada com o emprego de hardware e software livre. O uso de modens celulares para comunicação com os institutos de pesquisa diminui o tempo do envio dos alertas. O desenvolvimento de Web Services para o envio de dados torna o sistema mais seguro e de fácil implantação. Como a plataforma que coleta os dados é um sistema embarcado com poucos recursos de memória foi necessário o uso de um pequeno mais completo banco de dados relacional. Esta estrutura que tem um orçamento de implantação baixo e pode cobrir áreas sensíveis e tem como objetivo deixar muita mais rápida a coleta destes dados que podem contribuir para minimizar eventos extremos.*

Abstract. *This article describes a management system alerts in a hydrological services-based architecture. This structure is formed with the use of hardware and free software. The use of cellular modems for communication with the research institutes decreases the time of sending alerts. Developing Web Services for sending data makes the system more secure and easy to deploy. As the platform which collects the data is an embedded system with limited memory resources was necessary to use a little more complete relational database. This structure has a low-budget deployment and can cover sensitive areas and aims to make much more rapid collection of data that can help reduce extreme events.*

1. Introdução

Nos últimos anos vários tragédias ocorreram no país devido ao grande volume de chuvas, muitas destas ocorrências se devem ao aquecimento global e na maioria destes eventos extremos são difíceis de prever com mais de 24 horas de antecedência. Estas tragédias têm ocorrido com mais freqüência e levando a grandes perdas materiais e principalmente perdemos muitas vidas humanas.

Em sete anos e meio, o governo federal gastou cinco vezes mais com a reconstrução e assistência para as vítimas de desastres do que com a prevenção deles. Segundo o jornal Folha de S.Paulo, entre 2003 e junho deste ano, foram liberados R$ 5,8 bilhões para ações pós-tragédias e R$ 1,1 bilhão para prevenir enchentes como as que provocaram 45 mortes em cidades do Nordeste [Folha SP 2010].

As soluções para prevenir estas tragédias não atendem de forma completa as necessidades de sistemas de tempo real para mitigar estas ocorrências.

As plataformas de coleta de dados “PCDs” instaladas em vários pontos do território brasileiro são de dois tipos, as que transmitem seus dados por satélite (SCD e Cbers) e as que armazenam seus dados (DataLogger) e depois de um certo período estes dados são recuperados.

Estas plataformas não conseguem fazer alertas em tempo real, pois estas PCDs que transmitem seus dados via satélite brasileiro tem um atraso da transmissão, até o dado estar processado, de no mínimo 01 (uma) hora e meia, mas pode chegar ate 03 (três) horas, o que pode ser um tempo muito grande para um evento extremo.

As PCDs do tipo DataLogger não transmitem seus dados portanto não são indicadas para este tipo de aplicação.

A motivação para este trabalho é propor uma plataforma de coleta de dados usando software embarcado que após a aquisição dos dados coletados dos sensores e detectada alguma ocorrência fora dos limites estabelecidos para a área em questão, um alerta seja enviado ao Centro de Monitoramento mais próximo e com isso alguma ação seja efetuada em tempo hábil para evitar maiores transtornos para a população da região.

Esta proposta é baseada em uma tecnologia de Web Services para troca de dados via XML entre cliente e servidor e transmitindo seus dados por meio de uma rede de dados celular usando os protocolos GSM/GPRS/EDGE para o acesso a internet.

A facilidade e simplicidade desta plataforma têm muito a contribuir para facilitar a implantação desta solução em regiões propensas a sofrer eventos extremos.

2. Materiais e Métodos

2.1. Sistemas Embarcados

Nos últimos anos, os sistemas embarcados eram sempre caracterizados por dispositivos muito simples, com grandes restrições, como uso de memória,

desempenho do processador e sua bateria duravam muito pouco. Sua funcionalidade era determinada, tipicamente, via hardware, sendo que seu software usualmente era composto por *drivers* de dispositivos, escalonador de tarefas e alguma lógica de controle, causando uma baixa dinamicidade das tarefas ofertadas por um dado sistema ao longo do seu tempo de vida.

Recentemente, este cenário mudou bastante. Cada vez mais sistemas dispositivos que precisam de mais processamento, usam-se processadores ARM, chips RISC de 32 bits. Possuem um design simples, se comparado aos processadores x86, mas possuem bom desempenho híbridos, convergentes e não críticos, com inúmeras características dos sistemas de propósito geral estão surgindo.

Alguns Hardwares para o uso em aplicações embarcadas atuais evoluíram e são aplicados em na construção de computares pessoais os chamados NetBooks.

Com esta evolução existe a possibilidade de instalarmos softwares mais complexos com servidores de web e pequenos bancos de dados e sistemas desenvolvidos na linguagem Java.

Os System-on-a-chip (SoC) ou sistema-em-um-chip, se refere a todos os componentes de um computador, ou qualquer outro sistema eletrônico, em um circuito integrado (chip), ele pode conter funções digitais, analógicas [Nurmi 2009].

A Texas Instruments lançou recentemente um chip ARM ® Cortex -A8 que funciona a 1 GHz e tem 512 MB LPDDR RAM e pode rodar sistemas operacionais Linux embarcados e de tempo real de varias distribuições e Windows CE (figura 2).

2.2. Web Services

Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os Web services são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML [Hansen 2007].

Para as empresas, os Web services podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana.

Essencialmente, o Web Service faz com que os recursos da aplicação do software estejam disponíveis sobre a rede de uma forma normalizada. Outras tecnologias fazem a mesma coisa, como por exemplo, os browsers da Internet acedem às páginas Web disponíveis usando por norma as tecnologias da Internet, HTTP e HTML. No entanto, estas tecnologias não são bem sucedidas na comunicação e integração de aplicações. Existe uma grande motivação sobre a tecnologia Web Service pois possibilita que diferentes aplicações comuniquem entre si e utilizem recursos diferentes.

Utilizando a tecnologia Web Service, uma aplicação pode invocar outra para efetuar tarefas simples ou complexas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos.

Os Web Services são identificados por um URI (Uniform Resource Identifier), descritos e definidos usando XML (Extensible Markup Language). Um dos motivos que tornam os Web Services atrativos é o fato deste modelo ser baseado em tecnologias standards, em particular XML e HTTP (Hypertext Transfer Protocol). Os Web Services são utilizados para disponibilizar serviços interativos na Web, podendo ser acedidos por outras aplicações usando, por exemplo, o protocolo SOAP (Simple Object Access Protocol).

2.3. Redes de Transmissão de Dados GSM/GPRS/EDGE

O padrão GSM (Global System Mobile) foi concebido na década de 1980 pelo Conselho Europeu Telecommunication Standards Institute (ETSI), um sistema digital móvel que substituiria a telefonia analógica existente. O espírito da norma foi a de criar um sistema que pudesse transportar maior capacidade, tentando garantir uma unificação da telefonia móvel[Gome 2005].

O padrão GSM tem sido um sucesso nas tecnologias chamadas de 2G, sendo adotada por diversos países não-europeus em todo o mundo e até mesmo substituir, em alguns casos as redes 2G, como é o caso de alguns grandes operadores os E.U. e América Latina. Atualmente, cerca de GSM representa 80% das redes 2G do mundo.

GPRS (General Packet Radio Service): O Padrão de Transmissão de Rádio por Pacote (GPRS) é a evolução da tecnologia GSM em 2,5G. Essa tecnologia oferece velocidades máximas de dados de 115 kbps e um throughput médio de 30 a 40 kbps.

Enhanced Data rates for GSM Evolution (EDGE) ou Enhanced GPRS (EGPRS), é uma tecnologia digital para telefonia celular que permite melhorar a transmissão de dados e aumentar a confiabilidade da transmissão de dados. Embora o EDGE seja tecnicamente uma tecnologia da 3ª Geração, geralmente é classificada como um padrão 2,75G, já que é uma melhoria feita nas redes 2,5G (GPRS) e não a criação de um sistema propriamente dito. EDGE foi introduzido nas redes GSM no mundo por volta de 2003, inicialmente na América do Norte.

EDGE foi desenvolvida para capacitar a transmissão de uma grande quantidade de dados a altas taxas de velocidade (até 560 kbit/s). EDGE usa o mesmo conceito da tecnologia TDMA, no que se refere à estrutura dos quadros, canais lógicos e largura de banda.

2.4. Banco de Dados H2DB

O H2 é um sistema de gerenciamento de banco de dados relacional escrito em Java. Pode ser embutido em aplicações Java ou executar no modo cliente-servidor. Seu tamanho reduzido de aproximadamente 1MB se aplica perfeitamente a aplicações embarcadas.

O Banco de Dados é um subconjunto da SQL (Structured Query Language) padrão e ele têm suporte as principais APIs de programação SQL e JDBC, no entanto, o banco também oferece suporte usando o driver ODBC do PostgreSQL, agindo como um servidor PostgreSQL.

É possível criar tabelas na memória, bem como as tabelas baseadas em disco. As tabelas podem ser persistentes ou temporários. tipos de índice são tabela hash e tree para as tabelas em memória, e b-tree para tabelas baseadas em disco.

Todas as operações de manipulação de dados são transacionais. Assim como o bloqueio de tabelas e o controle de acesso simultâneo as mesmas.

O H2 tem como características de segurança, direitos de acesso à base de criptografia, a senha de uso SHA-256 e de dados usando o AES ou o Tiny Encryption Algorithm, XTEA. Os recursos de criptografia estão disponíveis em funções dentro do banco de dados. Conexões SSL / TLS são suportados no modo cliente-servidor, bem como ao usar o aplicativo de console.

3. Estudo de Caso

Para atender uma necessidade da Sociedade e da Defesa Civil desenvolvemos um modelo para alerta que tem um baixo custo e é baseado em software livre em uma arquitetura de Serviços Web (Web Services), visando diminuir o tempo de espera entre a aquisição de dados hidrometeorológicos por uma Plataforma de Coleta de Dados (PCD) e o envio de alertas aos Centros de Monitoramento para uma tomada de decisão, em caso de eventos extremos.

A PCD disponibiliza informações hidrometereológicas, visando um melhor acompanhamento das condições hidrológicas e uma melhor previsão de eventos extremos tais como secas, inundações, entre outros.

Cada PCD deve ser capaz de propiciar a transmissão dos dados coletados via um desses meios de comunicação previstos: celular, radiofreqüência ou satelital.

O Gerenciamento de alertas visa auxiliar os responsáveis quando um parâmetro sai dos limites estabelecidos.

Para a aplicação neste Estudo de Caso, considerou-se a seguinte arquitetura do sistema:

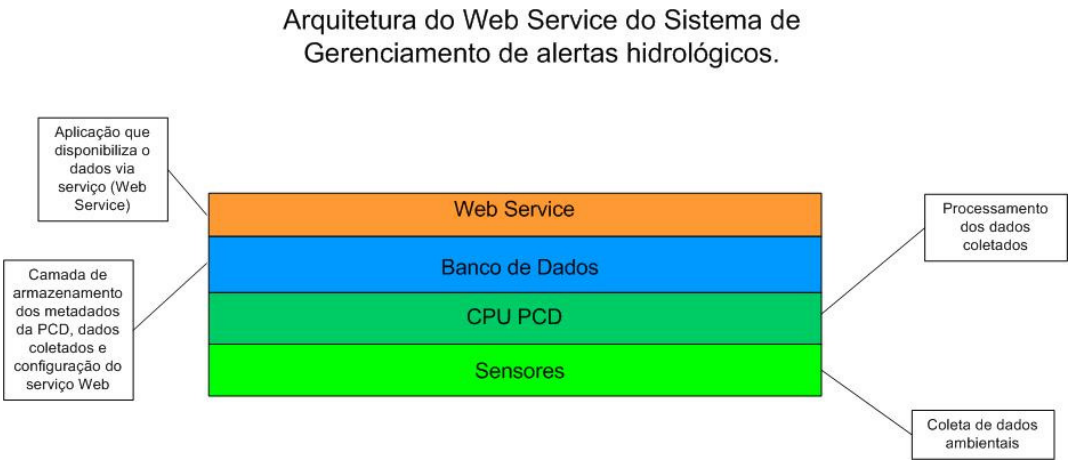


Figura 1 - Arquitetura do Sistema

3.1. Requisitos não funcionais do Sistema de Alerta

A Placa Gerenciadora/Processadora Micro-processada BeagleBoard (**Erro! Fonte de referência não encontrada.**) é uma placa eletrônica de baixo consumo e baixo custo desenvolvida pela Texas INC. e que tem seu projeto aberto (open source hardware).

A placa tem toda a funcionalidade de um computador básico. Ela usa o microprocessador ARM Córtex-A8 e pode rodar varias distribuições de Linux embarcadas de tempo real, onde serão instalados os softwares citados neste documento.

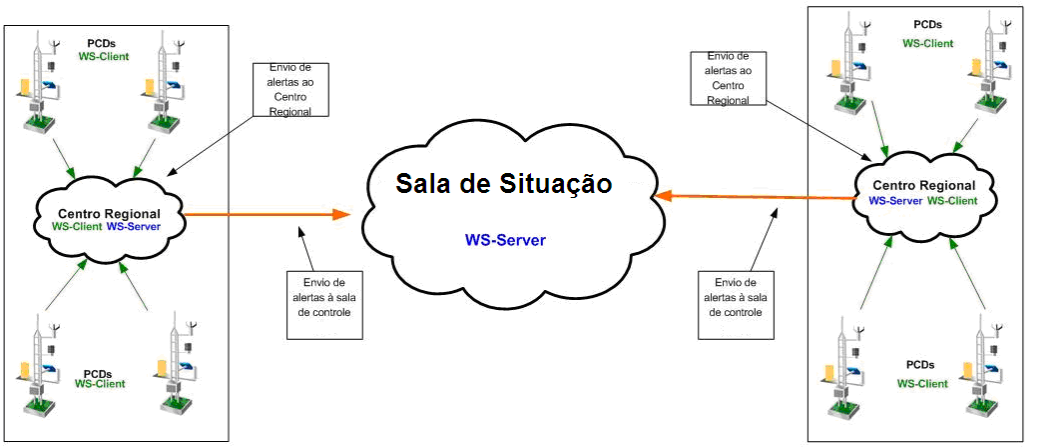
3.2. Modelo Proposto para um Sistema Embarcado

O modelo proposto para um **Si**stema de **G**erenciamento de **A**lertas (**SIGA**) deverá ser capaz de propiciar:

- Na camada cliente, o envio de alertas para os Centros de Monitoramento ou Sala de Situação, contendo na mensagem, a identificação da PCD, a data e hora da ocorrência, a variável (Sensor) e o valor do dado.

- Na camada servidora, o recebimento do alerta, a validação do conteúdo da mensagem, o armazenamento da ocorrência e a integração com o sistema de visualização e monitoramento das respectivas Salas de Controle.

Uma visão do SIGA é apresentada na Figura 2.



WS-Client – "Web Service Client" - Camada cliente do serviço, em execução na PCD e nos Centros Regionais, que envia os alertas para o Centro Regional e para a Sala de Controle

WS-Server – "Web Service Server" - Camada servidora do serviço, que recebem os alertas para a tomada de decisão nos Centros Regionais e no Centro Regional.

Figura 2 - Visão do SIGA

3.3. Banco de Dados.

O modelo implementado deve ser simplificado, com a única função de representar a configuração mínima, necessária e suficiente para a execução da aplicação cliente do SIGA. No que diz respeito a:

- **metadados da PCD**: código de identificação, latitude, longitude, altitude etc.
- **dados coletados**: data/hora da coleta, variável (Sensor) e valor.
- **configuração do serviço**: valores limites por sensor, tempo de execução, número de alertas por envio, tempo de armazenamento de alertas etc.

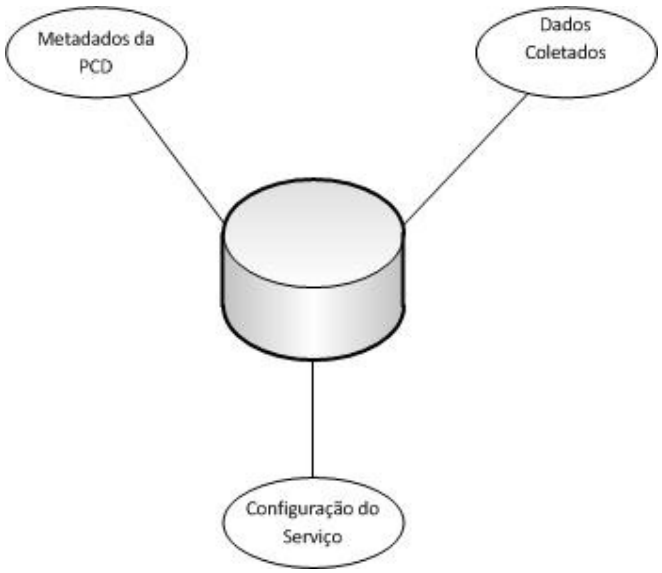


Figura 3 - Banco de Dados da PCD

3.4. Criação do Web Service

Para a criação do Web Service, que receberá os alertas enviados das PCDs, basta incluir a anotação @WebService na classe de serviço e a anotação @WebMethod nos métodos a serem disponibilizados. Estas anotações estão disponíveis na JSR 224: Java API for XML-Based Web Services (JAX-WS) 2.0 do JavaSE 1.6.
A classe de serviço:

```
package br.ita.sisatmh2.ws;
@WebService
public class AlertaMonitor {

  @WebMethod
  public String getAlerta(Integer pcd_id, String pcd_var, Float pcd_valor) {
    return "PCD - " + pcd_id + " Variavel - " + pcd_var + " Valor - " +
String.valueOf(pcd_valor);
            }
}
```

Para gerar o Web Service, executa-se a ferramenta apt (annotation processing tool) com o comando:

$apt br/ita/sisatmh2/ws/AlertaMonitor.java

Como resultado é gerado pacote br.ita.sisatmh2.ws.javaws com duas classes, uma classe GetAlerta.java para os parâmetros do WebService (no nosso caso, os parâmetros do Alerta) e outra GetAlertaResponse.java que representa o retorno.
A publicação do serviço é feita da seguinte maneira:

```
public class PublicaService {
```

```
 public static void main(String[] args){
   OiMundo service = new OiMundo();
   Endpoint endpoint = Endpoint.publish("http://localhost:8080/getAlerta", service);
  }
}
```

Onde:

- ⇨ **http://localhost:8080** – devem ser configurados para o endereço e porta do ambiente real do Centro de Monitoramento.

Pronto, o WebService está rodando sem servidor de aplicação nem web container. Para acessar a WSDL() dele, usa-se o URL:

http://localhost:8080/getAlerta?wsdl

3.5. Criação do Cliente para Web Service

Para criar o cliente do serviço, javaSE 6 já vem com as ferramentas necessárias. Utilizamos o **wsimport** (na pasta bin do jdk) para gerar as classes do cliente. Com o **serviço rodando**, executamos o comando:

```
$wsimport -keep -p br.ita.sisatmh2.ws.cliente http://localhost:8080/getAlerta?wsdl
```

Onde:

- ⇨ A opção ***-keep*** não apaga os arquivos fontes e ***-p*** gera as classes dentro do pacote especificado.
- ⇨ **br.ita.sisatmh2.ws**.cliente é o pacote destino das classes geradas.
- ⇨ **http://localhost:8080/getAlerta?wsdl** é o endereço de publicação do serviço, no caso em estudo, é o endereço do Centro de Monitoramento.

A classe AlertaMonitor que foi gerada pelo *wsimport* é uma **interface**, que é implementada pela classe AlertaMonitor em execução do lado do servidor.

Para chamar o serviço pelas classes geradas, escreva a seguinte classe dentro do pacote br.ita.sisatmh2.ws.cliente:

```
public class TestarServico {

  public static void main(String[] args) {
     br.ita.sisatmh2.ws.cliente.AlertaMonitor port = new AlertaMonitorService().getAlert
aMonitorPort();
  System.out.println(port.getAlerta(32000,"Pluviometro", 10.5));  }
}
```

3.6. Integração do Sistema

Como integração final, a aplicação cliente deve ser embarcada nas PCDs da rede de monitoramento e o Web Service, instalado e publicado no Centro de Monitoramento.

Na PCD, o banco de dados de ser populado com os metadados e a configuração do Web Service

4. Conclusão.

O objetivo deste trabalho é apresentar um Modelo para um Sistema de Gerenciamento de alertas hidrológicos em uma arquitetura baseada em serviços que visa atender uma necessidade da Sociedade e da Defesa Civil, deixando muita mais rápida e eficiente a tomada de decisão em caso de eventos extremos. O baixo custo e a simplicidade na implementação foram os principais fatores considerados neste trabalho.

Nós pretendemos em trabalhos futuros, implementar o uso de plataformas menores em torno desta atual e interligar as mesmas com o uso de redes sem fio.

5. Referências:

Ramos, D. B., Loubach, D. S., and da Cunha, A. M. (2010). Esqueletotipação: Um Método para Desenvolvimento de Software Embarcado.

Gomez G., Sanchez R. (2005). End-to-End Quality of Service over Cellular Networks, Editora Wiley.

Hansen D. (2007). SOA Using Java(TM) Web Services, Editora Prentice Hall

Yaghmour K. Masters J. Ben-Yossef G. Gerum P. (2008).Building Embedded Linux Systems, Editora O'Reilly Media

Nurmi J. (2009). Processor Design: System-On-Chip Computing for ASICs and FPGAs, Editora Springer Netherlands

JSR 224: Java™ API for XML-Based Web Services (JAX-WS) 2.0 (http://jcp.org/en/jsr/detail?id=224). Acessado em 07/12/2010.

WSDL - Web Services Description Language (http://www.w3.org/TR/wsdl). Acessado em 07/12/2010.