R U P I
(Robot Unified Platform Initiative)

2007. 3.

지능형로봇연구단

김 현

(hyunkim@etri.re.kr)

ETRI 한국전자통신연구원
Electronics and Telecommunications Research Institute

RUPI 개요

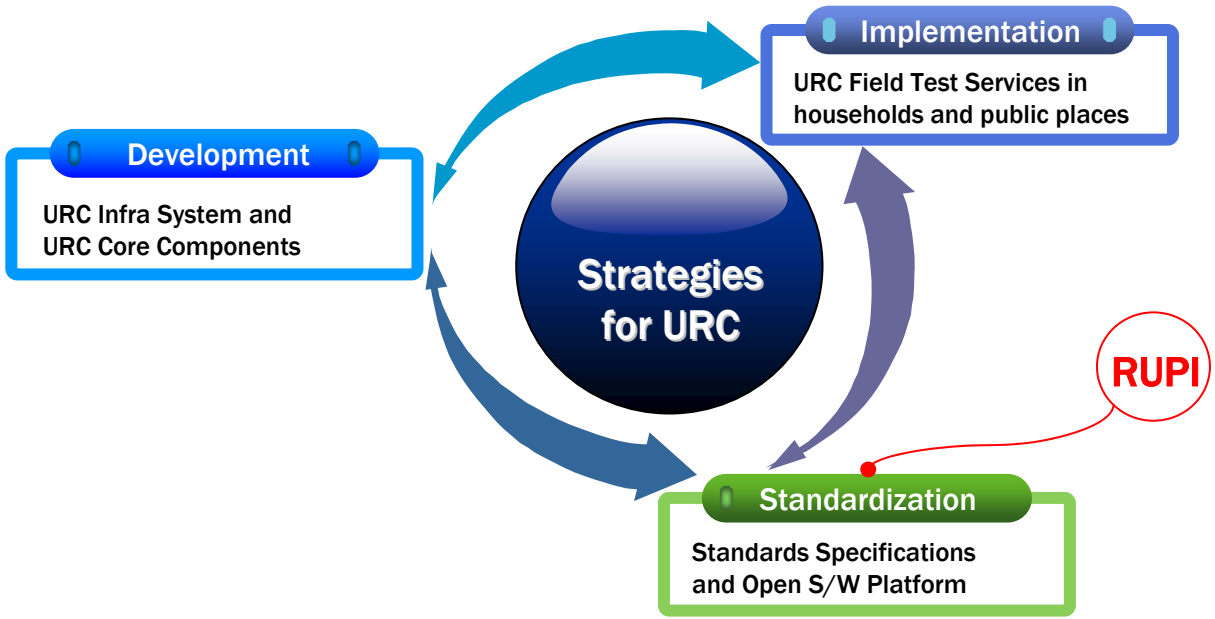
RUPI 개념 및 정의

RUPI (Robot Unified Platform Initiative)

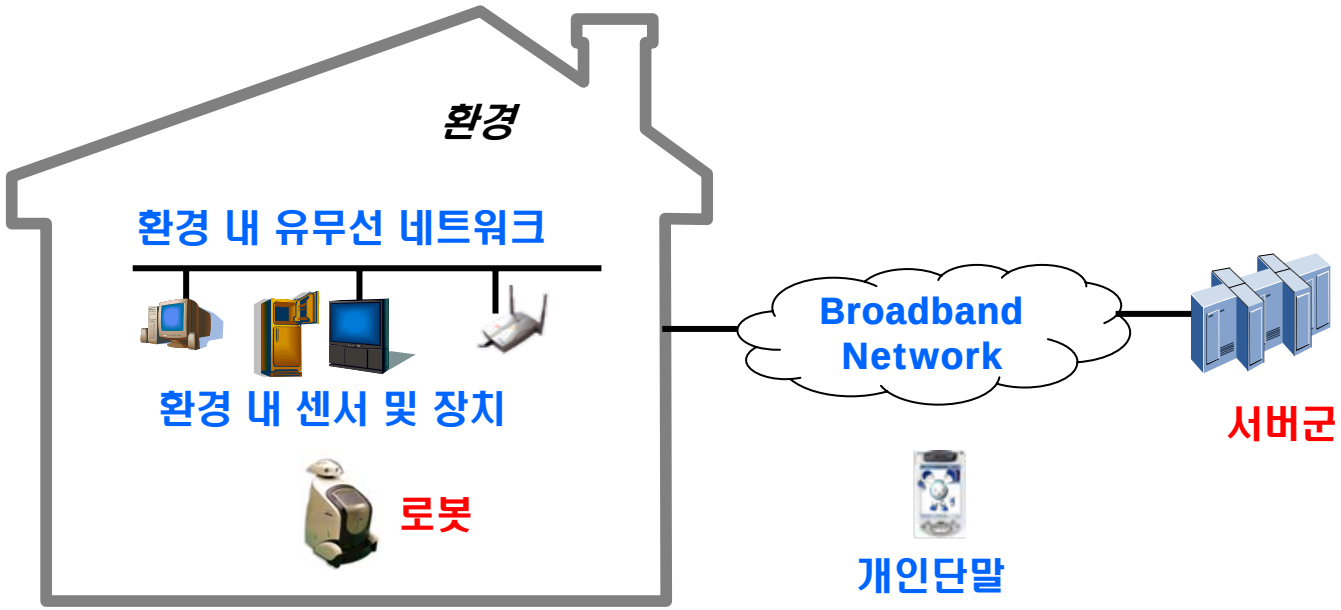
RUPI is an open standards for network based robots (URC)

다양한 로봇 플랫폼이 서버와 연동하여 다양한 로봇 서비스를 수행할 수 있도록 지원하는 네트워크 기반 로봇 (URC)의 표준 환경을 제공하는 제반 규격 및 플랫폼

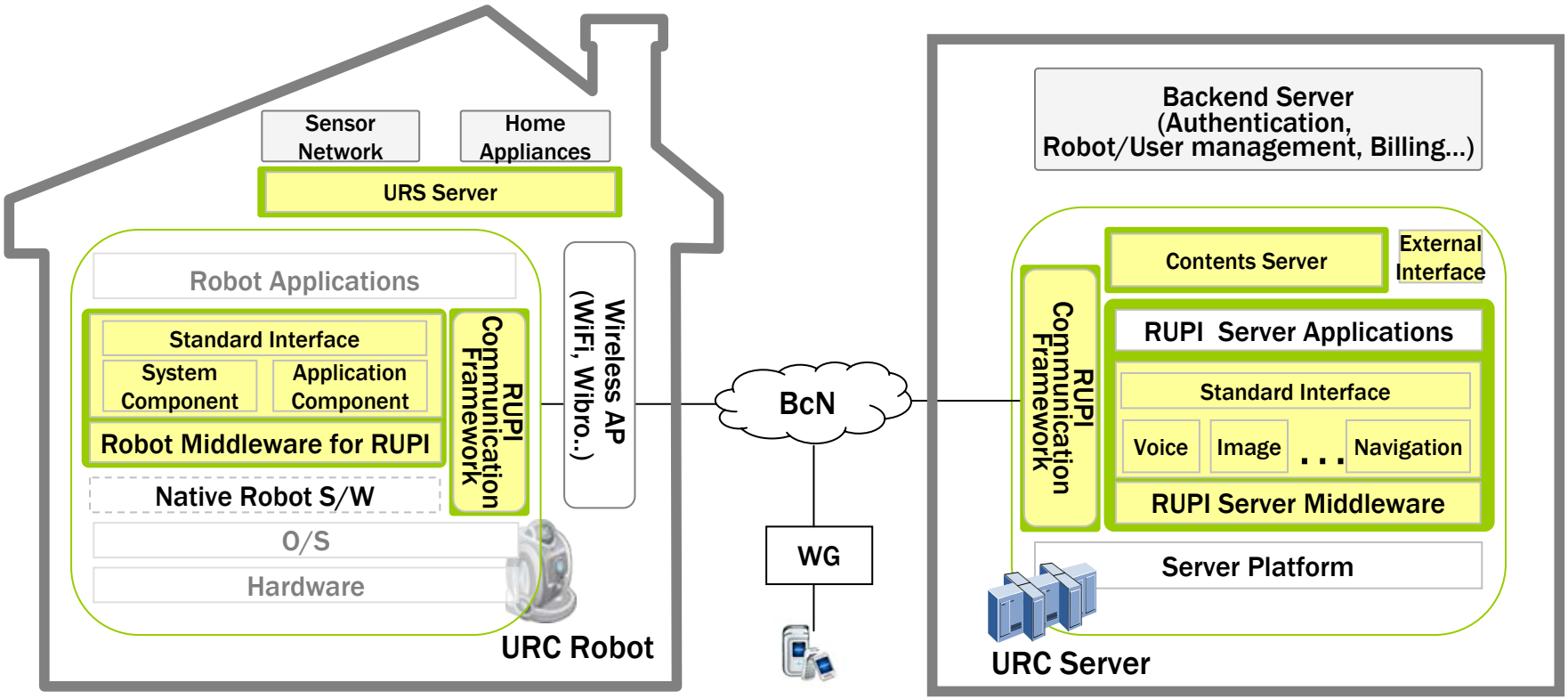
- 로봇 S/W 컴포넌트간의 상호호환성
- 다양한 통신 및 정보기기와의 상호운용성
- 이종 통신망과의 상호접속성



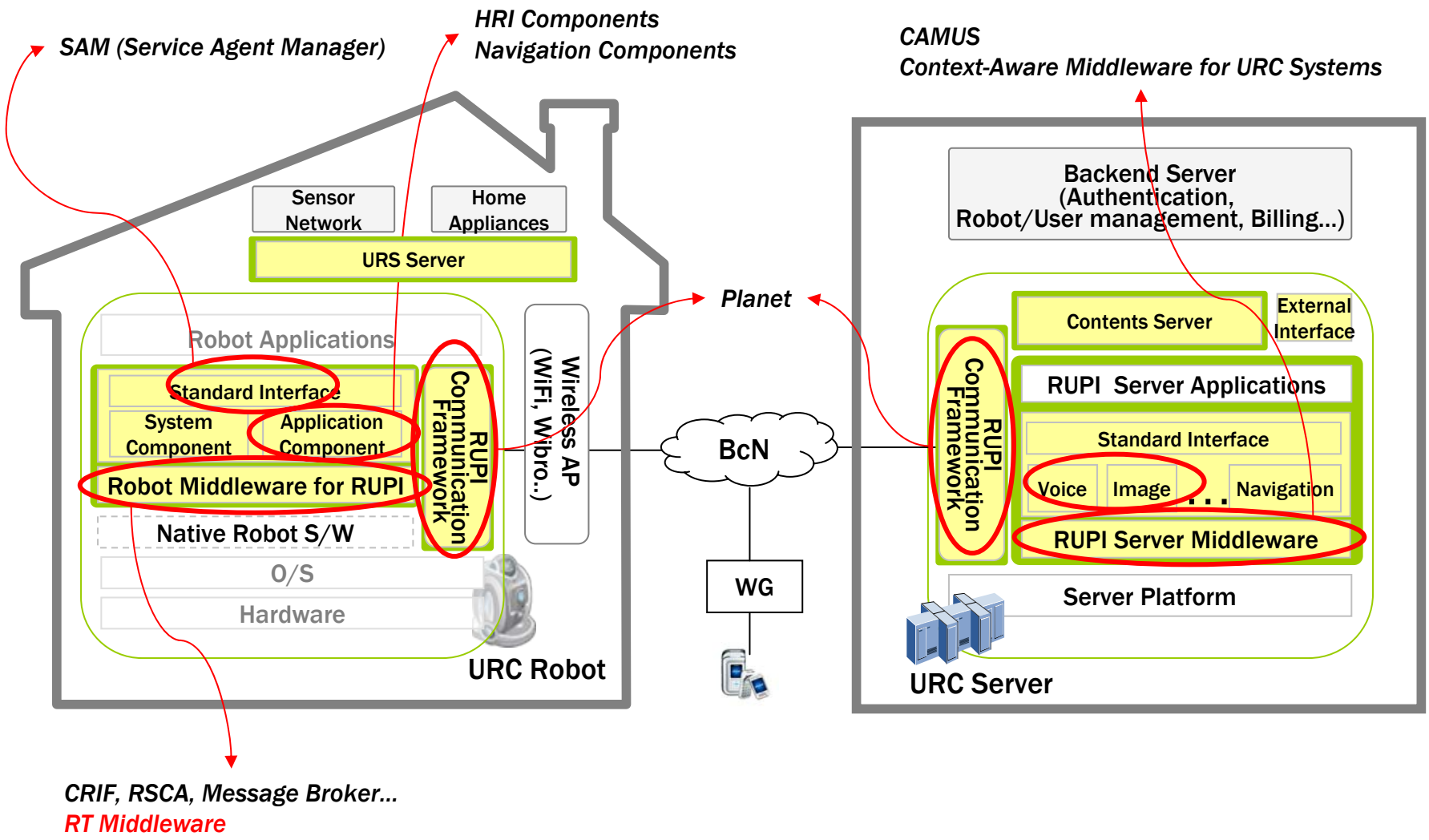
RUPI에 고려되어야 할 구성 요소



RUPI 개념적 구조


Sensor Network
Home Appliances
URS Server
Robot Applications
Standard Interface
System Component
Application Component
Robot Middleware for RUPI
Native Robot S/W
O/S
Hardware
RUPI Communication Framework
Wireless AP (WiFi, Wibro..)
URC Robot
BcN
WG
Backend Server (Authentication, Robot/User management, Billing...)
Contents Server
External Interface
RUPI Server Applications
Standard Interface
Voice
Image
...
Navigation
RUPI Server Middleware
Server Platform
RUPI Communication Framework
URC Server

RUPI 관련 기술 개발 현황



RUPI v.1.0
서버/로봇 간 통신 규격

Considerations

전제

- RUPI는 다양한 로봇과 다양한 서버들로 구성된 분산 시스템을 대상으로 한다.

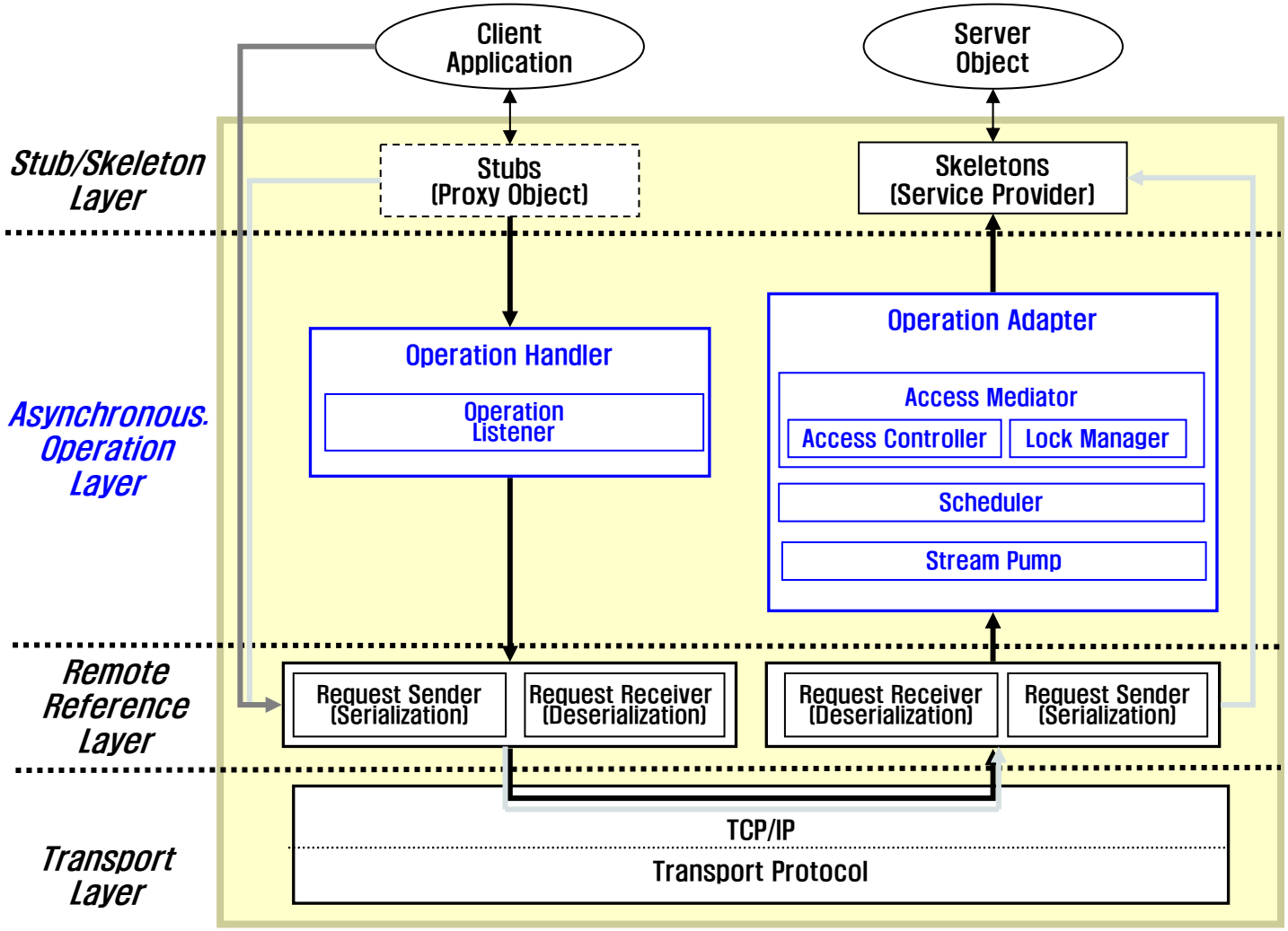
고려사항

- 특정 OS나 언어에 종속되어서는 안된다.
- 일반적인 컴퓨팅 환경뿐만 아니라 임베디드 시스템과 같이 상대적으로 자원이 제한된 장치의 환경에 대한 고려가 필요하다
- 음성, 영상 등 크기가 큰 데이터의 전송을 효과적으로 처리하기 위한 방안이 필요하다.
- 음성합성, 로봇 주행 등 비교적 긴 시간의 연산을 처리할 수 있어야 한다.
- 비교적 불안전한 무선 통신을 사용하기 때문에 통신 단절에 대한 대처가 고려되어야 한다.

비고

- 관련 기술: DCOM, CORBA, RMI, XML Web Services

통신 프레임워크 구조


Client Application
Server Object
Stub/Skeleton Layer
Stubs (Proxy Object)
Skeletons (Service Provider)
Operation Handler
Operation Listener
Operation Adapter
Access Mediator
Access Controller
Lock Manager
Scheduler
Stream Pump
Asynchronous Operation Layer
Remote Reference Layer
Request Sender (Serialization)
Request Receiver (Deserialization)
Request Receiver (Deserialization)
Request Sender (Serialization)
TCP/IP
Transport Protocol
Transport Layer

데이터 인코딩 (1/4)

자료형	인코딩 (Backus-Naur Form)
boolean	<boolean> ::= 0x01 | 0x02 # (True | False)
byte	<byte> ::= b8 # 8bit
short	<short> ::= b16 b8 # 16bit, big-endian
int	<int> ::= b32 b24 b16 b8 # 32bit, big-endian
long	<long> ::= b64 b56 ... b8 # 64bit, big-endian
float	<float> ::= b32 b24 b16 b8 # IEEE floating number
double	<double> ::= b64 b56 ... b8 # IEEE floating number
string	<string> ::= <int:length> b*length #UTF-8 encoded
binary	<binary> ::= <int:length> b*length #대량byte열(BLOB)
array	<array> ::= <int:length><typed:element>*length
farray	<farray> ::= <int: length > <data:elementi>*length #final array
list	<list> ::= <int:length> <typed:element>*length
map	<map> ::= <int:n> (<typed:key> <typed:value>) *n
enum	<int:ordinal> #나열 ordinal 값을 <int>형식으로 인코딩

데이터 인코딩 (2/4)

자료형	인코딩 (Backus-Naur Form)
type	<type> ::= <byte:type_code> # 기본 타입 | (0x15 <string:type_name>) # 확장형 타입 | ((0xOD|0xOE) <type:elm_type>) # array | (0x1A <string:enum_type_name>) # enum | (0x1C <throwable>) # throwable | 0x00 # null
valued	<valued>::=<int:field_length>(<typed:field$_i$>)*field_length # 사용자 정의 객체 <tvalued>::=0x17 <string:type_name> <valued>
remote	<remote> ::= <string:host> <int:port> <string[]:path> # 원격객체 reference <tremote> ::= 0x18 <string:remote_type> <remote>
throwable	<throwable> ::= <string:exception_name> <string:details> # 예외 <tthrowable> ::= 0x1C<throwable>
appdef	<appdef> ::= any <tappdef> ::=0x1B <string:encoding-tag><appdef>

데이터 인코딩 (3/4)

- InputStream
  - 송신자 측에서 planet.connection.RemoteInputStream 객체 생성 후, 그것의 참조를 전달

```
<instream> ::= <string:host>      # callee 측 호스트 주소
               <int:port>         # callee 측 port
               <string[]:path>    # 생성된 RemoteInputStream servant의 path
               <string:type>      # planet.connection.RemoteInputStream
```

  - 수신자자 측에서는 전달된 객체의 'getBytes' 메소드를 호출하여 일정 크기의 binary 데이터를 가져옴

```
package planet.connection;
public interface RemoteInputStream {
    ByteBuffer getBytes(int size) throws IOException;
    void close() throws IOException;
}
```

데이터 인코딩 (4/4)

- 비동기 연산 (Asynchronous Operation) 처리
  - 비동기 연산 수행상태 모니터링
  - 비동기 연산 제어
  - 비동기 연산 호출

```
@PlanetInterface
public interface TextToSpeech {
    public AsyncOperation speak(String text, String speaker);
    public void cancelAll();
    public String[] getAvailableSpeakers();
}
```

```
TextToSpeech tts = …;

AsyncOperation aop = tts.speak(“안녕하세요”, “미진”);
aop.waitForCompleted();

aop = tts.speak(“또 안녕하세요”, “아람”);
Thread.sleep(1000);
aop.cancel(true);
```

메시지 인코딩 (1/5)

- 메시지 구조
  - 연결요청 (connect)
  - 메시지 헤더 (header)
  - 함수 호출 (call)
  - 결과값 전달 (reply)
  - Error 전달 (error)

메시지 인코딩 (2/5)

- connect
  - 소켓 연결 후 연결 요청자가 처음으로 전달하는 메시지로서, 한 Planet Server가 다른 Planet Server와 연결을 맺기 위해 전달하는 메시지
  - host: 연결을 요청하는 Planet Server의 IP 주소
  - port: 연결을 요청하는 Planet Server의 포트 번호

```
<connect> ::= <string:host> <int:port>
```

메시지 인코딩 (3/5)

- header
  - 모든 메시지 앞에 붙는 헤더 정보
  - magic: Planet 메시지 임을 표시
  - request_id: 요청 식별자. (매 call 메시지마다 단조 증가)
  - length: 메시지 길이 (헤더 포함)
  - code: 메시지 종류
    - 100: call
    - 101: reply
    - 102: error
    - 103: notify

```
<header> ::=    <int:magic>          // magic number(970208)
                <int:request_id>     // 요청 식별자
                <int:length>         // 메시지 길이
                <byte:code>          // 메시지 타입
```

메시지 인코딩 (4/5)

- **call**
  - 지정된 원격 객체 연산을 호출하기 위해 전달되는 메시지

```
<call> ::=   <header>                    // 메시지 헤더
             <string[]:path>             // 대상 servant의 path
             <type:declaring_type>       // 호출 메소드가 속한 타입
             <string:method_name>        // 호출 메소드 이름
             <int:arg_count>             // 호출 인자 개수
             <typed:arg>*arg_count       // 호출 인자 값들
```

- **reply**
  - call 메시지 요청에 의해 수행된 결과를 전달하기 위한 메시지

```
<reply> ::= <header>                     // 메시지 헤더
            <type:declared_result_type>  // 메소드에 선언된 반환 타입
            <typed:result>               // 실제 반환 값
```

메시지 인코딩 (5/5)

- error
  - 원격 호출의 결과로 예외가 발생된 경우, reply 대신 회신

```
<error> ::= <header>                    // 메시지 헤더
            <throwable:exception>       // 예외 객체
```

- notify
  - 인코딩은 <call>과 동일
  - 요청 후 reply를 대기하지 않음

```
<notify> ::=  <call>
```

메시지 인코딩 – Heartbeat

- Heatbeat
  - <notify> 메시지 이용

```
<notify> ::= <header>                    // 메시지 헤더
             <string[]:path>             // “planet”
             <type:declaring_type>       // “planet.RemotePlanet”
             <string:method_name>        // “ping”
             <int:arg_count>             // 1
             <typed:arg>*arg_count       // <remote>
```

```
<notify> ::= <header>                    // 메시지 헤더
             <string[]:path>             // “planet”
             <type:declaring_type>       // “planet.RemotePlanet”
             <string:method_name>        // “pong”
             <int:arg_count>             // 0
             <typed:arg>*arg_count       // null
```

PLANET v.1.0

사양

- 다양한 언어 지원 : Java, C/C++
- 다양한 운영체제 지원 : Win32, Unix/Linux(Embedded Linux)

기능

- TCP (Socket) 기반의 로봇/서버 간 원격 객체 통신 기능
- Binary message encoding 방식에 따른 Light-weight protocol
- 효율적인 비동기 연산 (Asynchronous Operation) 지원
- 연산 Queuing 기능 제공
- 우수한 Local Transparency 지원
- CORBA 80% 내외 수준의 RTT (Round Trip Time)

관련 결과

- 국내 대부분 URC 로봇 적용

  삼성전자(에니봇, 청소로봇), 국민로봇(아이로비, 제니보, 네토로, 로보이드 등), 모스트아이텍, 이디, KIST 마루/아라 등
- TTA 표준기고서 2건 제출
- 국제특허 1건, 국내특허 1건 제출

RUPI v.1.0
서버/로봇 통신을 위한 표준 인터페이스

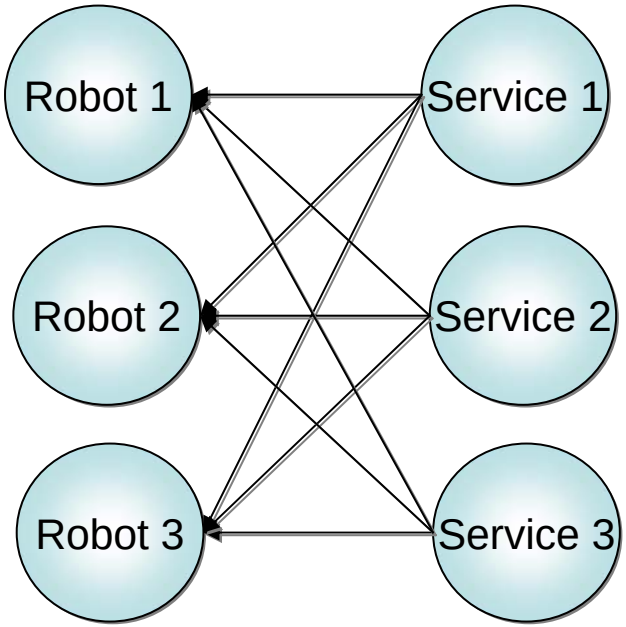
Considerations

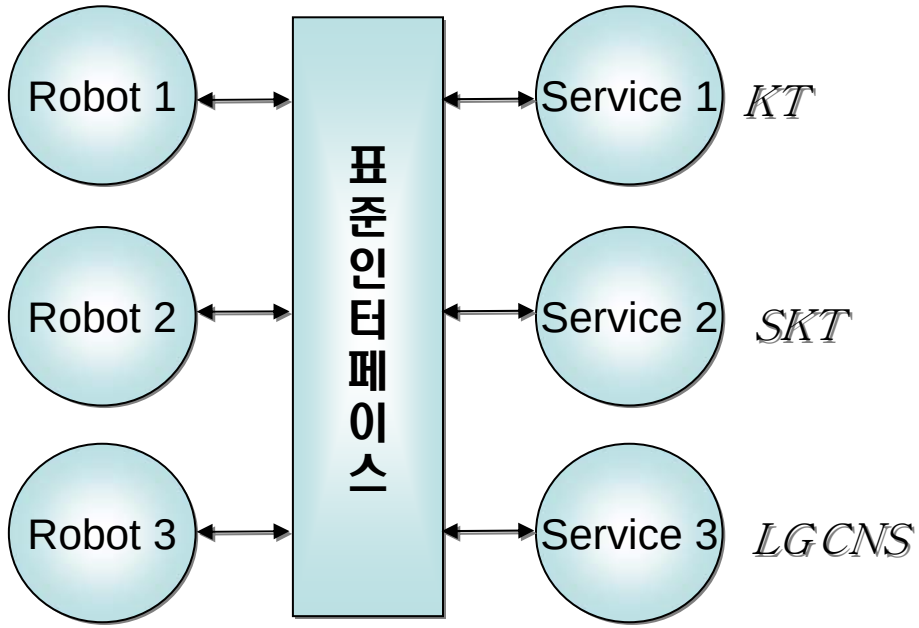
전제

- RUPI는 다양한 서비스가 다양한 로봇에게 제공된다.

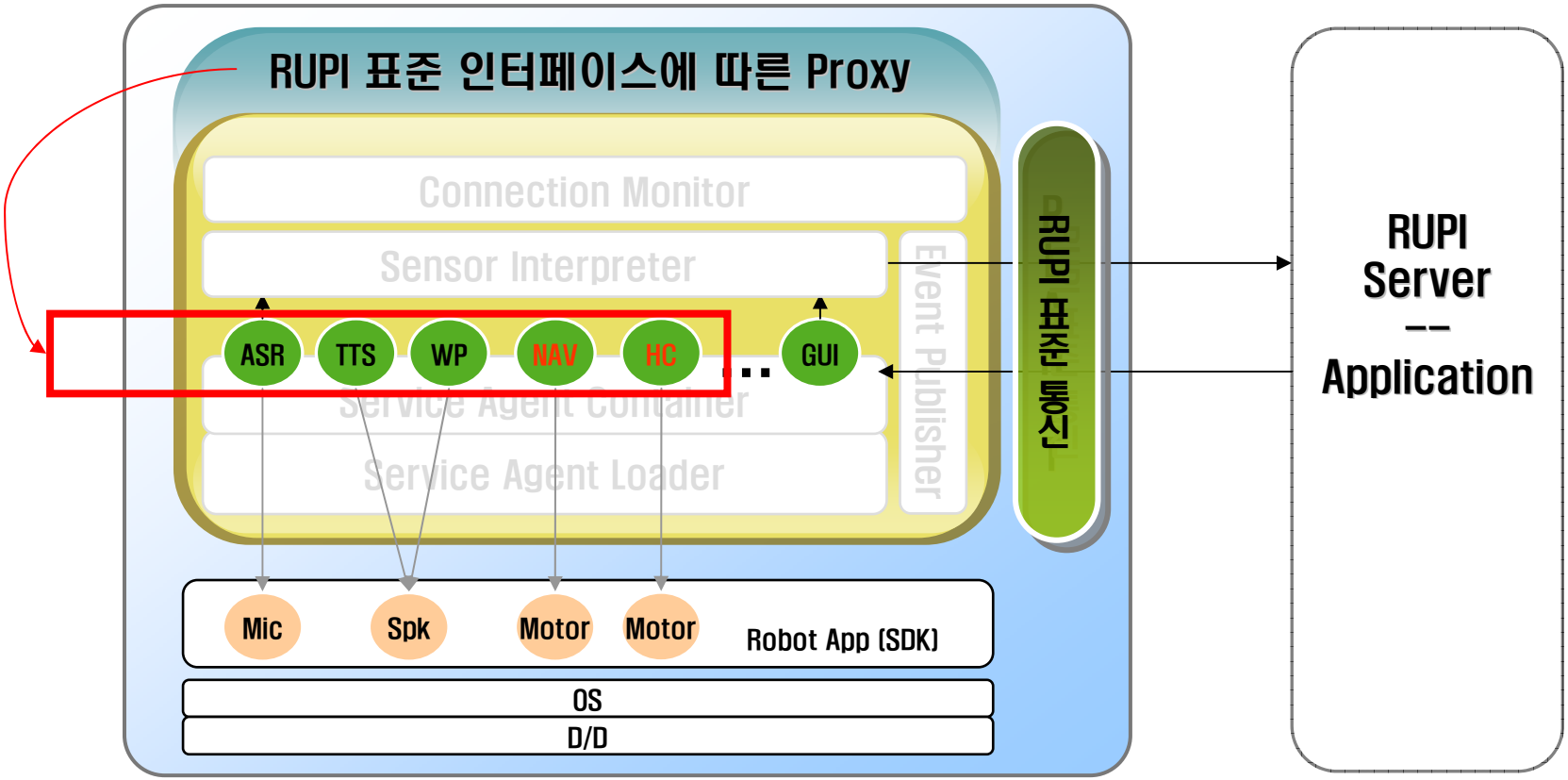
고려사항

- RUPI 통신 규약에 기반한다.
- RUPI 로봇의 최소한의 기능에 해당되는 부분을 표준화한다.
- 상속 또는 확장을 통해 부가적 기능을 추가할 수 있도록 한다.





시스템 개념


RUPI 표준 인터페이스에 따른 Proxy
Connection Monitor
Sensor Interpreter
Event Publisher
ASR
TTS
WP
NAV
HC
GUI
Service Agent Container
Service Agent Loader
RUPI 표준 통신
Mic
Spk
Motor
Motor
Robot App (SDK)
OS
D/D
RUPI
Server
--
Application

Service Agent: Example

```
public interface SpeechSensor {
   public void activate();
   public void deactivate();

   public void setSpeechList(SpeechList speechList);
   …
}


public interface TextToSpeech {
   public AsyncOperation speak(String text,
                               String speaker);
   public void stop(String details);
   public String[] getAvailableSpeakers();
}

public interface Navigation {
   public NavigationMap getNavigationMap();

   public AsyncOperation navigate(PathSegment[] path);
   public AsyncOperation navigateToCharger();
   public void stop (String details);

   public BodyPosture getPosition();
}
```

Standard Interface for SA

- 오디오 관련
  - 마이크: Microphone
  - 음성 합성: TextToSpeech, TextToWave
  - 음성 인식: SpeechSensor
  - 음원 재생: WavePlayer, MP3Player
- 영상 관련
  - 카메라: Camera
  - 인식: MotionSensor, FaceSensor
  - 재생: AVIPlayer (mpeg 포함)
- 로봇 관련
  - 움직임: MoveWheel, Navigation
  - 머리: MoveHead
  - 터치센서: TouchSensor
  - 기타: RobotNativeService
- 위치
  - RFIDSensor

Service Agent Manager v.1.0

사양

- 로봇/서버 간 표준 인터페이스 준수
- 다양한 OS 지원 : Win32, Unix/Linux (Embedded Linux)
- 시스템 최소 사양: CPU 200MHz, Memory 20Mb
  (SAM/PLANET: 5.9Mb)

기능

- CAMUS와 타 시스템/장치 간의 연동 G/W 역할
- 서버와의 통신을 위한 로봇 플랫폼 측에서의 Service Agent 관리
- Sensor Interpreter (로봇 측 센서 정보 Filtering 및 Aggregation)
- Event Publisher (로봇 측 Event의 Server 전달)
- Connection Monitor (서버와의 연결 상태 모니터링 및 연결 복구)

관련 결과

- KT, 삼성전자, LG CNS 등 국내 대기업에서 운영 중

  삼성전자(에니봇, 청소로봇), 국민로봇(아이로비, 제니보, 네토로, 로보이드 등), 모스트아이텍, 이디, KIST 마루/아라 등 대부분 URC 로봇 연동
- TTA 표준기고서 1건 제출
- 국제특허 1건 제출

감사합니다