TinyOS

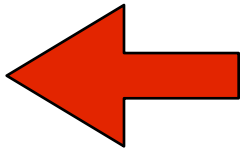
Hands-on Session

Goals

1. Install TinyOS
2. Layout of **`tinyos-2.x`**
3. Write two applications
   - (A) DisseminationDemoClient
   - (B) CollectionsDemoClient

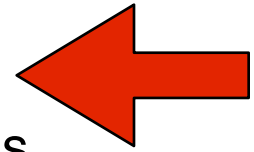
Options

- LiveCD
  - XubunTOS
  - Customized Ubuntu 8.10 LiveCD



Today

- Native
  - Linux
    - .rpm packages
    - .deb packages
  - Windows: Cygwin + .rpm packages
  - MacOS X
    - stow
    - macports



Recommended

Other Options

- VMware
  - Jetos
    - based on JeOS (Ubuntu Server 8.04)
    - optimized for ssh access
    - very small: 190MB compressed
  - Lenny
    - based on Debian 5.0 “Lenny”
    - graphical interface using XFCE
    - bigger: 300MB compressed
  - XubunTOS

Components

- NesC: nesc_*.deb
- Cross compiler
  - binutils: msp430-binutils-tinyos_*.deb
  - gcc: msp430-gcc-tinyos_*.deb
  - libc: msp430-libc-tinyos_*.deb
  - gdb (optional)
- Deputy: deputy-tinyos_*.deb

TinyOS

Environment

```
export TOSROOT=$HOME/local/src/tinyos-2.x
export TOSDIR=$TOSROOT/tos

export MAKERULES=$TOSROOT/support/make/Makerules

export CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar:.
export PYTHONPATH=$TOSROOT/support/sdk/python
```

Architectures

- AVR
  - mica2, mica2dot
  - micaz
  - btnode
  - IRIS
- ARM
  - imote2
- MSP430
  - telosb, sky
  - shimmer
  - eyesIFX
  - tinynode
  - epic
- 8051
  - CC2430
  - CC1110/CC1111

Layout

```
+ tinyos-2.x
   + apps
   + docs
   + support
   + tools
   + tos
```

Layout

```
+ apps
  + Blink
  + Null
  + RadioCountToLeds
  + MultihopOscilloscope
  + tests
    + ...
  + ...
+ docs
+ support
+ tools
+ tos
```

TinyOS

Layout

```
+ apps
+ docs
   + html
   + pdf
   + txt
   + ...
+ support
+ tools
+ tos
```

TinyOS

Layout

```
+ apps
+ docs
+ support
  + make
    - Makerules
    + avr/
    + msp/
    + ...
  + sdk
+ tools
+ tos
```

TinyOS

Layout

```
+ apps
+ docs
+ support
   + make
   + sdk
      + c
      + cpp
      + java
      + python
+ tools
+ tos
```

TinyOS

Layout

```
+ support
  + sdk
    + c
      + blip
      + sf
    + cpp
      + sf
    + java
      - tinyos.jar
    + python
      + tinyos
      - tos.py
```

TinyOS

Layout

```
+ apps
+ docs
+ support
+ tools
+ tos
   + chips
   + interfaces
   + lib
   + platforms
   + sensorboards
   + systems
   + types
```

TinyOS

Layout

```
+ tos
  + chips
    + atm128
    + msp430
    + pxa27x
    + cc2420
    + cc1000
    + at45db
    + stm25p
    + sht11
    + ...
```

TinyOS

Layout

```
+ tos
  + chips
  + interfaces
    - Boot.nc
    - SplitControl.nc
    - StdControl.nc
    - ...
  + lib
  + platforms
  + sensorboards
  + systems
  + types
```

Layout

```
+ tos
  + lib
    + net
    + printf
    + timer
    + tosthreads
    + serial
      - SerialActiveMessageC.nc
      - SerialAMSenderC.nc
      - SerialAMReceiverC.nc
      - ...
    + ...
```

Layout

```
+ tos
  + lib
    + net
      + ctp
      + 4bitle
      + drip
      + Deluge
      + dip
      + blip
      + ...
```

TinyOS

Layout

```
+ tos
  + systems
    - AMReceiverC.nc
    - AMSenderC.nc
    - MainC.nc
    - LedsC.nc
    - TimerMilliC.nc
    - ...
```

Layout

```
+ tos
   + chips
   + interfaces
   + lib
   + platforms
   + sensorboards
   + systems
   + types
      - TinyError.h
      - messsage.h
      - ...
```

Applications

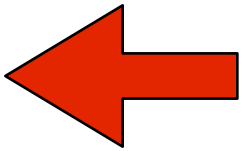
DisseminationDemo

CollectionDemo

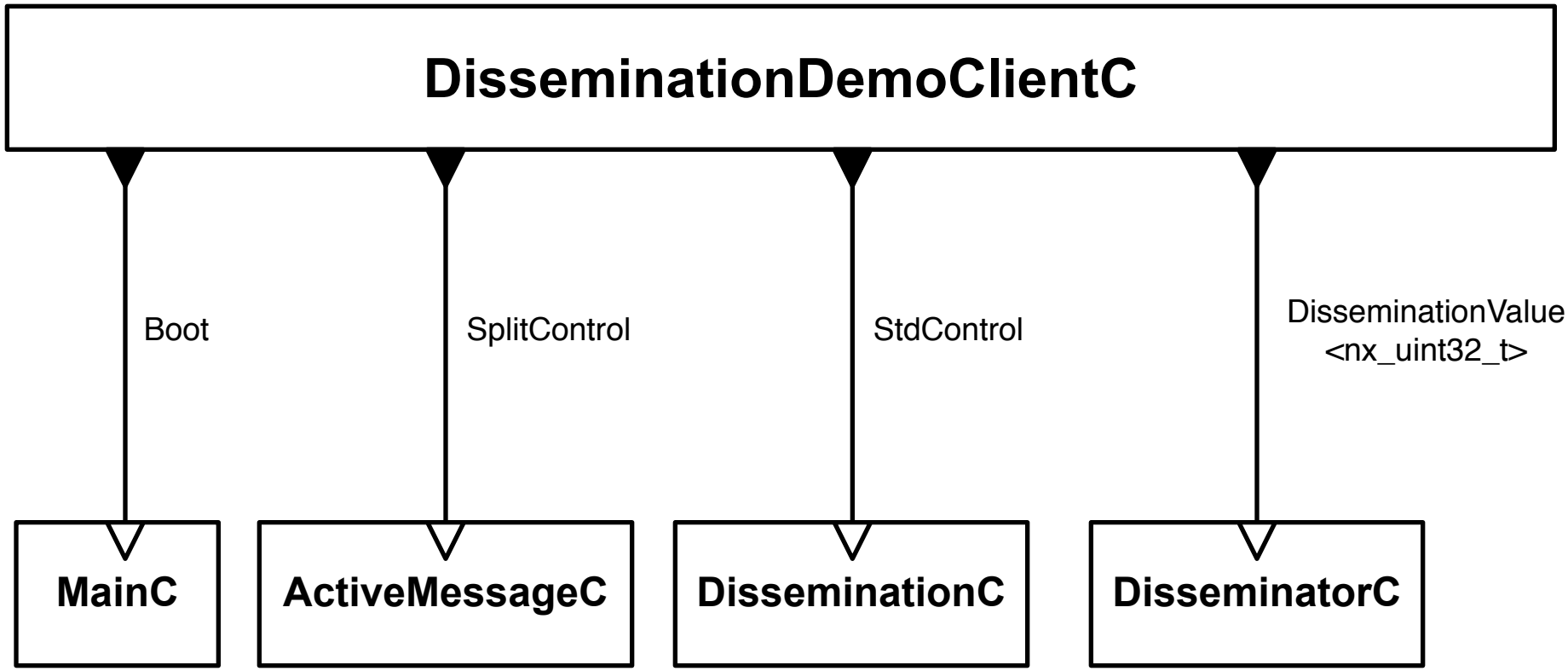
DisseminationDemo

DisseminationDemo

- DisseminationDemoClient
  - start the radio
  - start Drip
  - when a new value is received print its contents
- DisseminationDemoServer
  - start the radio
  - start Drip
  - start a periodic timer
  - on each firing or the timer increment a counter and disseminate it



DisseminationDemoClient



DisseminationDemoClient

- Interfaces
  - Boot
  - StdControl
  - SplitControl
  - DisseminationValue<t>
- Components
  - MainC
  - ActiveMessageC
  - DisseminationC
  - DisseminatorC

tos/interfaces/Boot.nc

```
interface Boot {
  event void booted();
}
```

tos/interfaces/StdControl.nc

```
interface StdControl
{
  command error_t start();
  command error_t stop();
}
```

tos/interfaces/SplitControl.nc

```
interface SplitControl
{
  command error_t start();
  event void startDone(error_t error);

  command error_t stop();
  event void stopDone(error_t error);
}
```

tos/lib/net/DisseminationValue.nc

```
interface DisseminationValue<t> {
  command const t* get();
  command void set(const t*);
  event void changed();
}
```

tos/system/MainC.nc

```
configuration MainC {
  provides interface Boot;
  uses interface Init as SoftwareInit;
}

implementation {
  ...
}
```

tos/platforms/telosa/ActiveMessageC.nc

```
configuration ActiveMessageC {
  provides {
    interface SplitControl;
    ...
  }
}

implementation {
 ...
}
```

tos/lib/net/drip/DisseminationC.nc

```
configuration DisseminationC {
  provides interface StdControl;
}

implementation {
  ...
}
```

tos/lib/net/drip/DisseminatorC.nc

```
generic configuration DisseminatorC(typedef t,
                                    uint16_t key) {
  provides interface DisseminationValue<t>;
  provides interface DisseminationUpdate<t>;
}

implementation {
  ...
}
```

Makefile

```
COMPONENT=DisseminationDemoClientAppC

CFLAGS += -I%T/lib/net
CFLAGS += -I%T/lib/net/drip
CFLAGS += -I%T/lib/printf

include $(MAKERULES)
```

Commands

```
$ make telosb

$ make telosb install,42

$ tos-dump.py serial@/dev/ttyUSB0:115200
```

Summary

```
tos/interfaces/Boot.nc
tos/interfaces/StdControl.nc
tos/interfaces/SplitControl.nc

tos/system/MainC.nc
tos/platforms/telosa/ActiveMessageC.nc
tos/lib/net/drip/DisseminationC.nc
tos/lib/net/drip/DisseminatorC.nc
```

DisseminationDemoClientAppC.nc

```
configuration DisseminationDemoClientAppC { }

implementation
{
  components MainC;
  components DisseminationC;
  components new DisseminatorC(nx_uint32_t, 2009);
  components DisseminationDemoClientC;
  components ActiveMessageC;

  DisseminationDemoClientC.Boot -> MainC;
  DisseminationDemoClientC.DisseminationStdControl -> DisseminationC;
  DisseminationDemoClientC.DisseminationValue -> DisseminatorC;
  DisseminationDemoClientC.RadioSplitControl -> ActiveMessageC;
}
```

DisseminationDemoClientC.nc

```
module DisseminationDemoClientC
{
  uses {
    interface Boot;
    interface DisseminationValue<nx_uint32_t>;
    interface StdControl as DisseminationStdControl;
    interface SplitControl as RadioSplitControl;
  }
}

implementation
{
  nx_uint32_t counter;

  event void Boot.booted()
  {
    call RadioSplitControl.start();
  }

  ...

}
```

DisseminationDemoClientC.nc

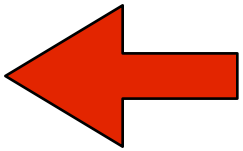
```
module DisseminationDemoClientC
{
   ...
}

implementation
{

  ...

  event void RadioSplitControl.startDone(error_t error)
  {
    call DisseminationStdControl.start();
  }

  event void DisseminationValue.changed()
  {
    printf("R: %lu\n", *(call DisseminationValue.get()));
    printfflush();
  }

  event void RadioSplitControl.stopDone(error_t error) { }
}
```

http://docs.tinyos.net/index.php/Ipsn2009-tutorial

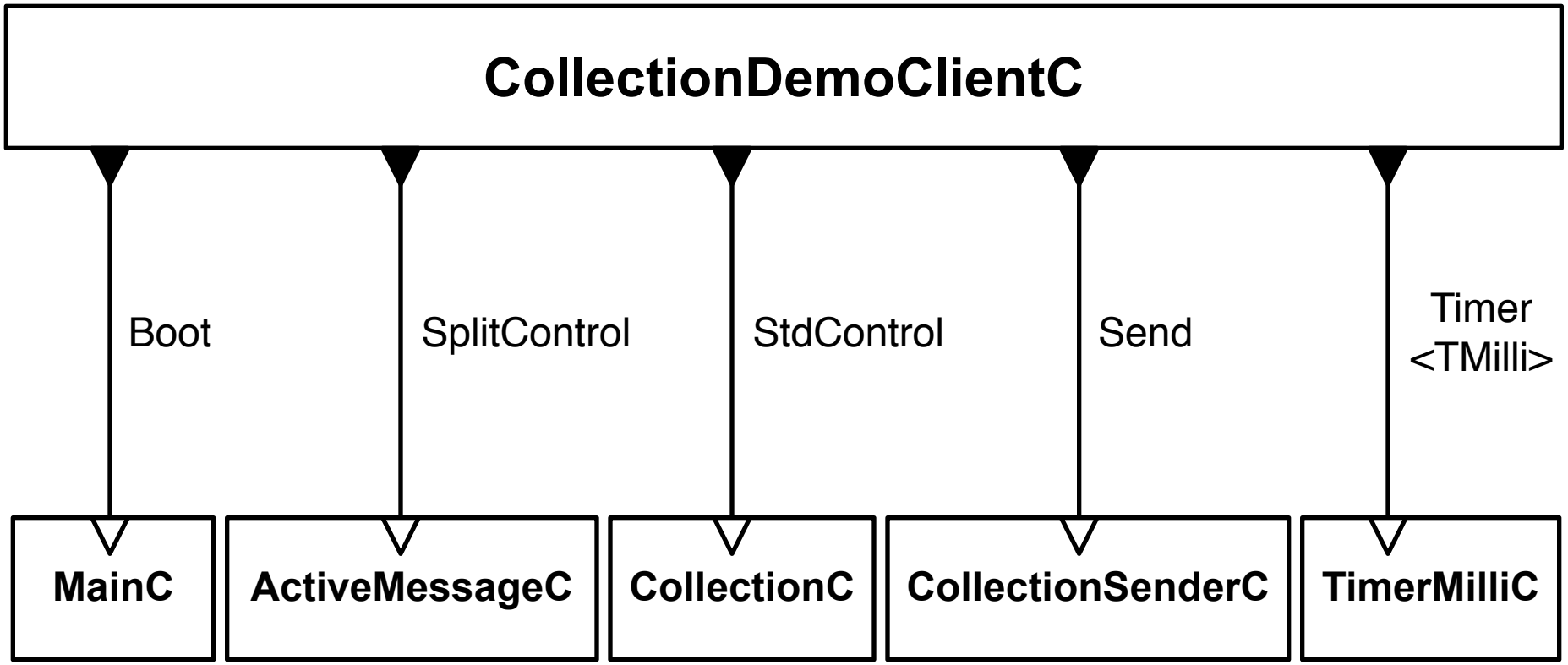
CollectionDemo

CollectionDemo

- CollectionDemoClient
  - start the radio
  - start CTP
  - start a periodic timer
  - on each firing or the timer increment a counter and sent it over CTP
- CollectionDemoServer
  - start the radio
  - start CTP
  - when a new value is received print its contents



CollectionDemoClient



CollectionDemoClient

- Interfaces
  - Boot
  - StdControl
  - SplitControl
  - Send
  - Timer<TMilli>
- Components
  - MainC
  - ActiveMessageC
  - CollectionC
  - CollectionSenderC
  - TimerMilliC

CollectionDemoClient

- Interfaces
  - Boot
  - StdControl
  - SplitControl
  - **Send**
  - **Timer<TMilli>**
- Components
  - MainC
  - ActiveMessageC
  - **CollectionC**
  - **CollectionSenderC**
  - **TimerMilliC**

tos/interfaces/Send.nc

```
interface Send {
  command error_t send(message_t* msg, uint8_t len);
  event void sendDone(message_t* msg, error_t error);
  command uint8_t maxPayloadLength();
  command void* getPayload(message_t* msg, uint8_t len);

  command error_t cancel(message_t* msg);
}
```

tos/lib/net/ctp/CollectionC.nc

```
configuration CollectionC {
  provides {
    interface StdControl;
    ...
  }
}

implementation {
  ...
}
```

tos/lib/net/ctp/CollectionSenderC.nc

```
generic configuration
CollectionSenderC(collection_id_t collectid) {
  provides {
    interface Send;
    interface Packet;
  }
}

implementation {
 ...
}
```

tos/system/TimerMilliC.nc

```
generic configuration TimerMilliC() {
  provides interface Timer<TMilli>;
}

implementation {
  ...
}
```

Makefile

```
COMPONENT=CollectionDemoClientAppC

CFLAGS += -I%T/lib/net
CFLAGS += -I%T/lib/net/ctp
CFLAGS += -I%T/lib/net/4bitle
CFLAGS += -I%T/lib/printf

include $(MAKERULES)
```

Summary

```
tos/interfaces/Boot.nc
tos/interfaces/StdControl.nc
tos/interfaces/SplitControl.nc
tos/interfaces/Send.nc
tos/lib/timer/Timer.nc

tos/system/MainC.nc
tos/system/TimerMilliC.nc
tos/platforms/telosa/ActiveMessageC.nc
tos/lib/net/ctp/CollectionC.nc
tos/lib/net/ctp/CollectionSenderC.nc
```

CollectionDemoClientAppC.nc

```
configuration CollectionDemoClientAppC { }

implementation
{
  components MainC;
  components ActiveMessageC;
  components CollectionC;
  components new CollectionSenderC(16);
  components new TimerMilliC() as Timer;
  components CollectionDemoClientC;

  CollectionDemoClientC.Boot -> MainC;
  CollectionDemoClientC.RadioSplitControl -> ActiveMessageC;
  CollectionDemoClientC.CollectionStdControl -> CollectionC;
  CollectionDemoClientC.Send -> CollectionSenderC;
  CollectionDemoClientC.Timer -> Timer;
}
```

CollectionDemoClientC.nc

```
module CollectionDemoClientC
{
  uses {
    interface Boot;
    interface SplitControl as RadioSplitControl;
    interface StdControl as CollectionStdControl;
    interface Send;
    interface Timer<TMilli>;
  }
}

implementation
{
  message_t smsg;

  typedef nx_struct {
    nx_uint8_t string[8];
    nx_uint16_t counter;
  } name_t;
  name_t *name;

  ...
}
```

CollectionDemoClientC.nc

```
module CollectionDemoClientC
{
  ...
}

implementation
{

  ...

  event void Boot.booted()
  {
    name = call Send.getPayload(&smsg, sizeof(name_t));
    strcpy((char*)name->string, "name");
    name->counter = 0;
    call RadioSplitControl.start();
  }

  ...

}
```

CollectionDemoClientC.nc

```
module CollectionDemoClientC
{
  ...
}

implementation
{

  ...

  event void RadioSplitControl.startDone(error_t error)
  {
    call CollectionStdControl.start();
    call Timer.startPeriodic(1024);
  }


  ...

}
```

CollectionDemoClientC.nc

```
module CollectionDemoClientC
{
  ...
}

implementation
{

  ...

  event void Timer.fired()
  {
    error_t error;
    name->counter++;
    error = call Send.send(&smsg, sizeof(name_t));
    printf("S: %d %d\n", name->counter, error);
    printfflush();
  }

  event void Send.sendDone(message_t* msg, error_t error) { }
  event void RadioSplitControl.stopDone(error_t error) { }
}
```

http://docs.tinyos.net/index.php/Ipsn2009-tutorial

The End.