
Network Reprogramming

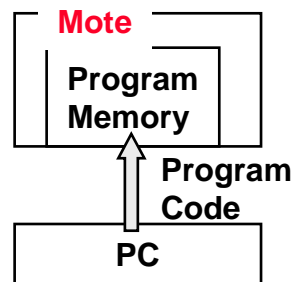
Jaein Jeong, Sukun Kim

Nest Meeting

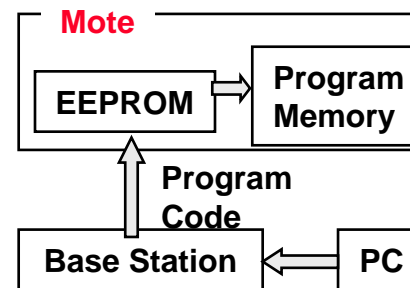
July 30, 2003

Concept

- **Parallel programming**
 - A mote is loaded program with direct connection
- **Network programming**
 - The program code is stored outside the program memory through radio packets.
 - Downloaded code is transferred to the program memory afterwards.



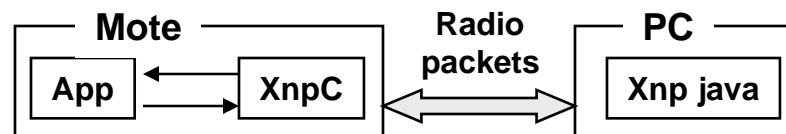
Parallel programming.



Network programming

Components

- **On motes**
 - XnpM module handles most of the jobs.
 - Main application needs to be wired to XnpC.
- **On PC**
 - Xnp java program sends program code through radio.
- **Message Formats**
 - AMID: set to #47.
 - CMD: type of command (e.g., download, query and etc).
 - PID: Checksum for program code. Used for validation
 - CID: Sequence number for capsule



0:1	2	3	4	5	6	7:8	9:10	11:end
Dest	AMID	GID	Len	CMD	SUBCMD	PID	CID	Data

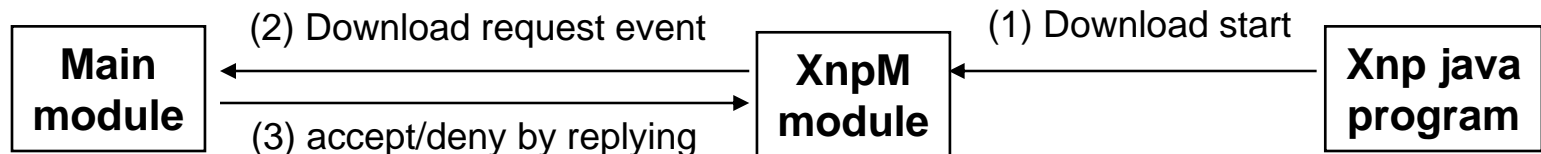
Steps of network programming

- **Download Phase**
- **Query Phase**
- **Reprogram Phase**

Download Phase

- **Start of download**

- Xnp java program sends a request and XnpM relays this request to the main module.
- Main module can reserve resources and acknowledges to XnpM.
- ‘Download start’ message is sent multiple times.

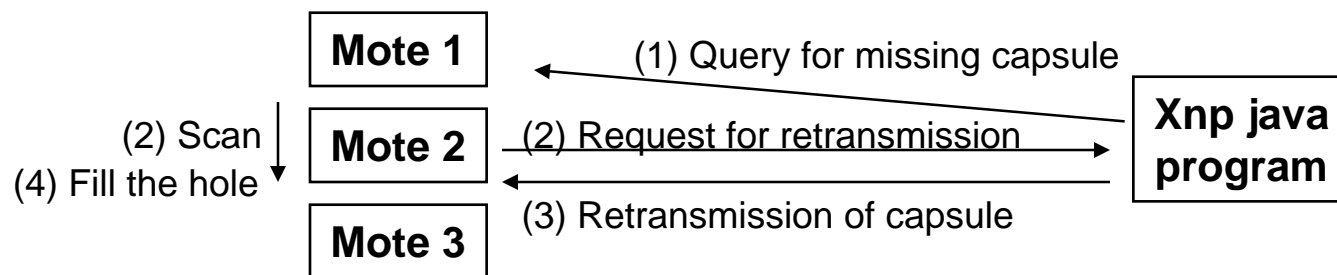


- **Download**

- java program sends each line of program as a capsule.
- The capsule is stored in EEPROM by XnpM.

Query Phase

- **A mote gets any missing capsules in query loop.**
 1. java program asks motes for missing capsules.
 2. Each mote scans EEPROM and requests the retransmission of the next missing capsule.
 3. In response, java program sends the missing capsule.
 4. Other motes can also fill the hole as well as the requestor.
- **Query loop ends when the java program doesn't get request for several times.**

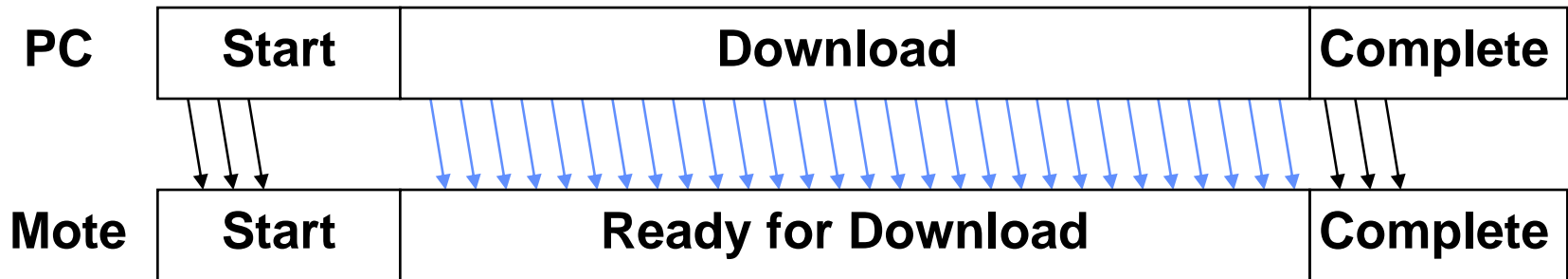


Reprogram Phase

- **The downloaded code is transferred to the program memory and the mote starts the new program.**
 - First, java program sends a reprogram request.
 - After checking the EEPROM for correctness, XnpM transfers control to the boot loader.
 - The boot loader copies the code in EEPROM to program memory and reboots the system.
- **After reboot, mote ID and group ID needs to set correctly.**

Protocol

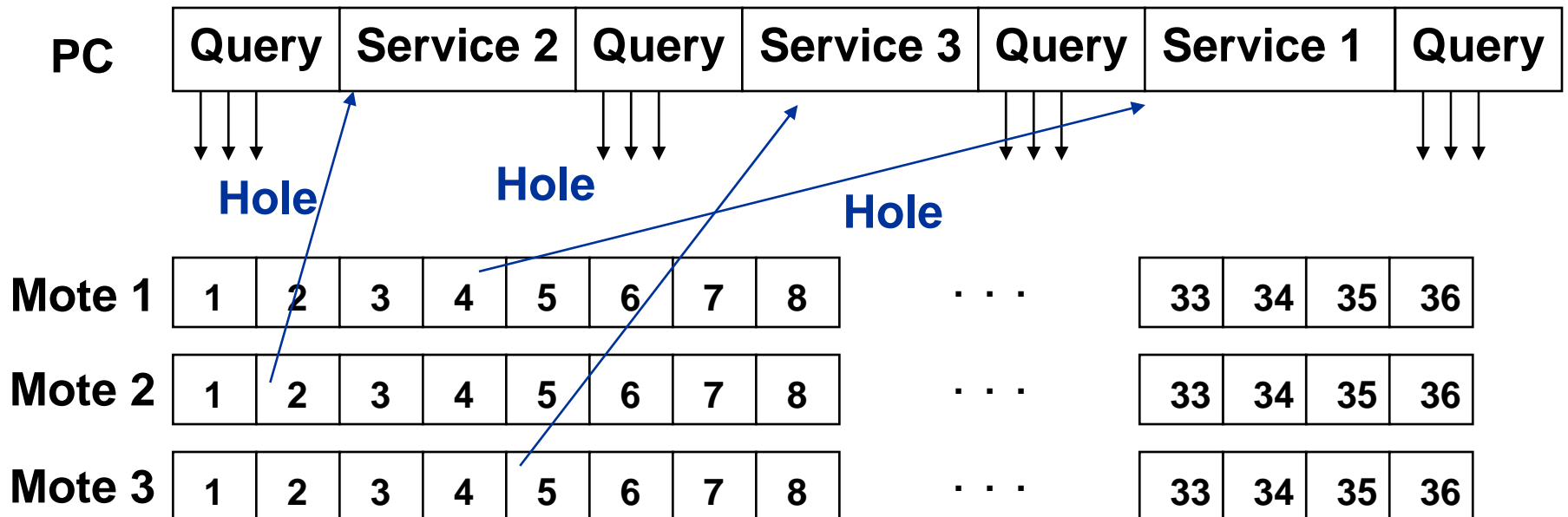
Download Phase



Query Phase

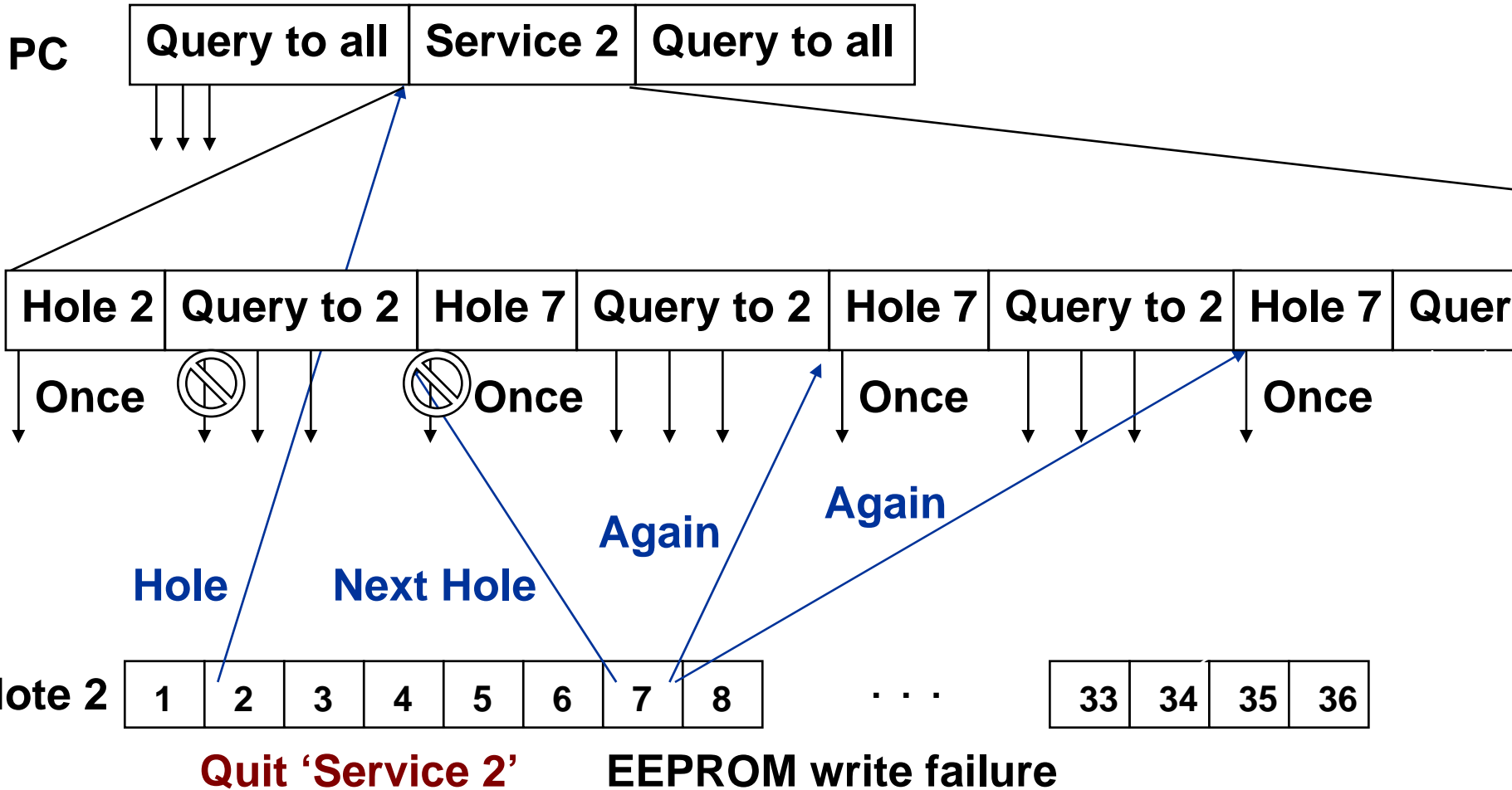
Completed !!!

Timeout



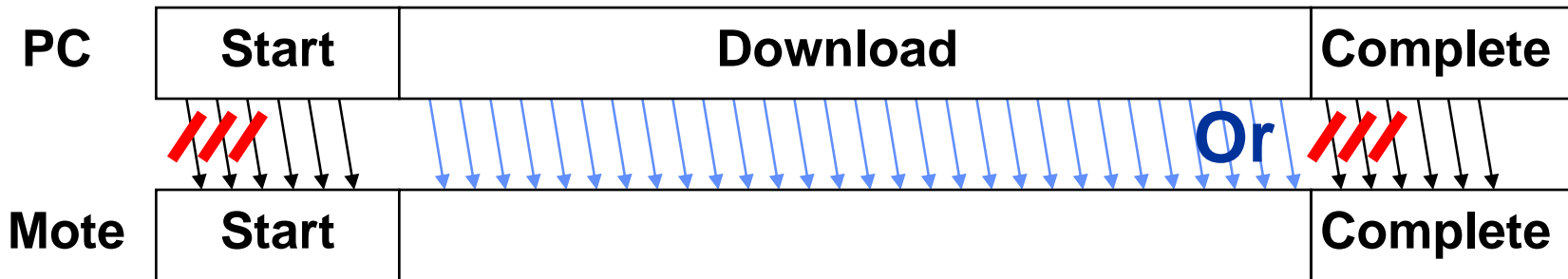
More detail in Phase 2

Query Phase



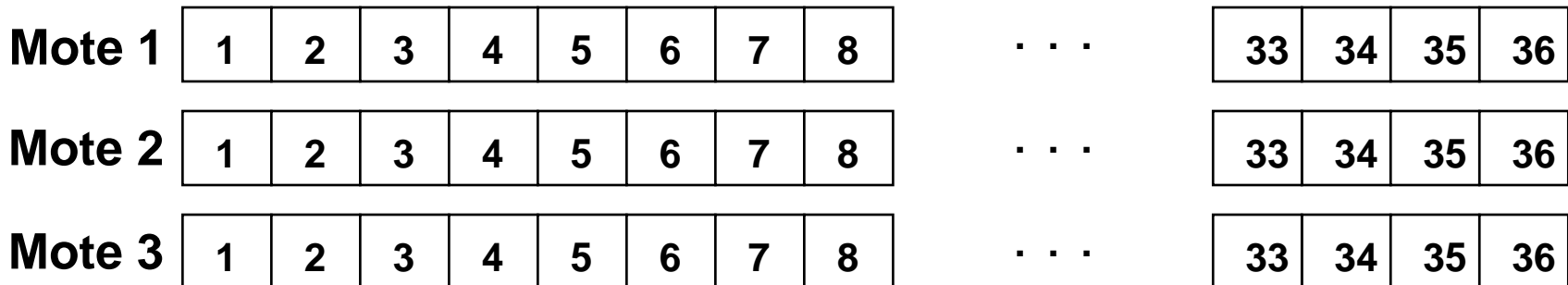
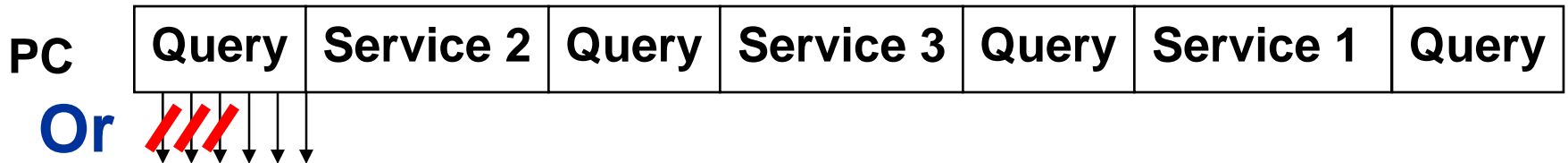
Modification Increase the number of control message retransmission

Download Phase



Failure !!!

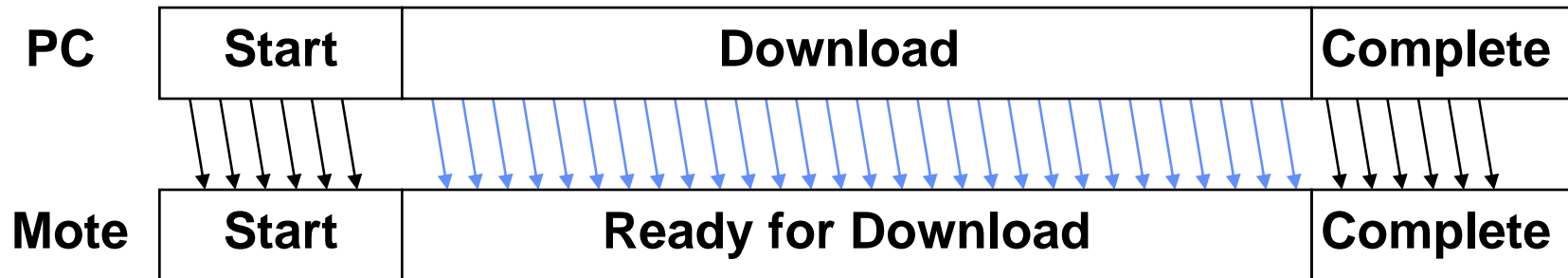
Query Phase



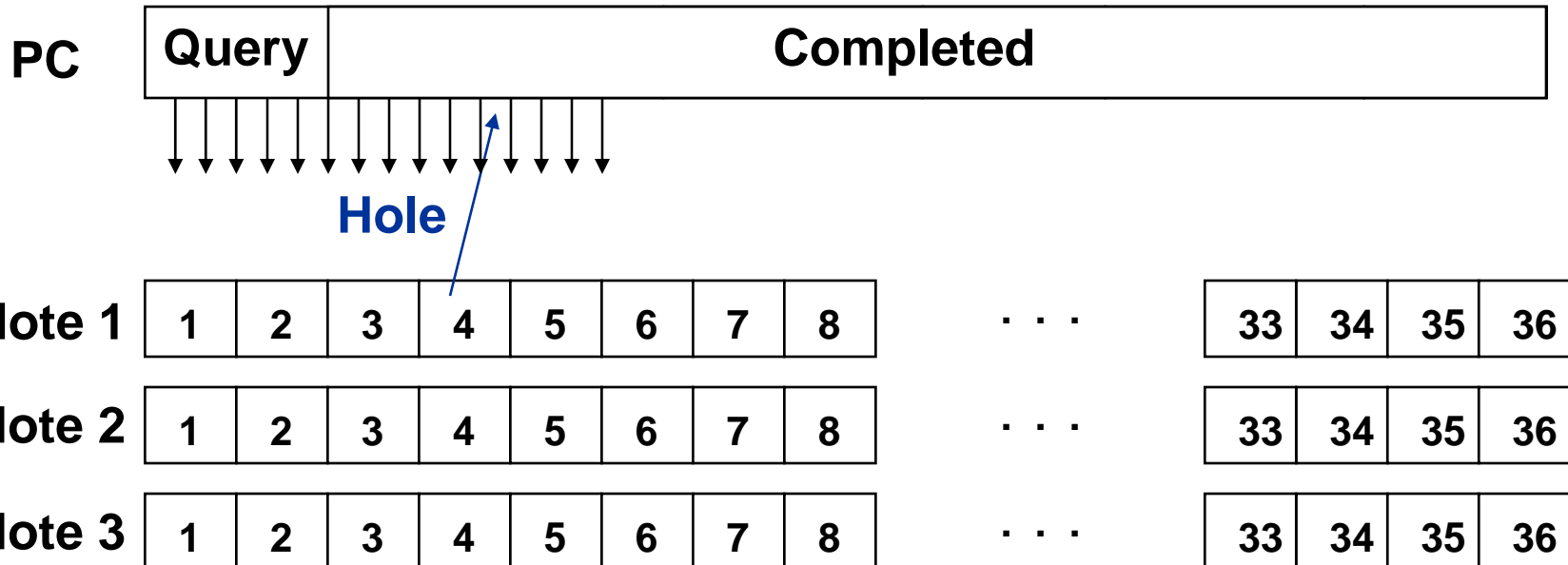
Increase query timeout

Modification (continued)

Download Phase

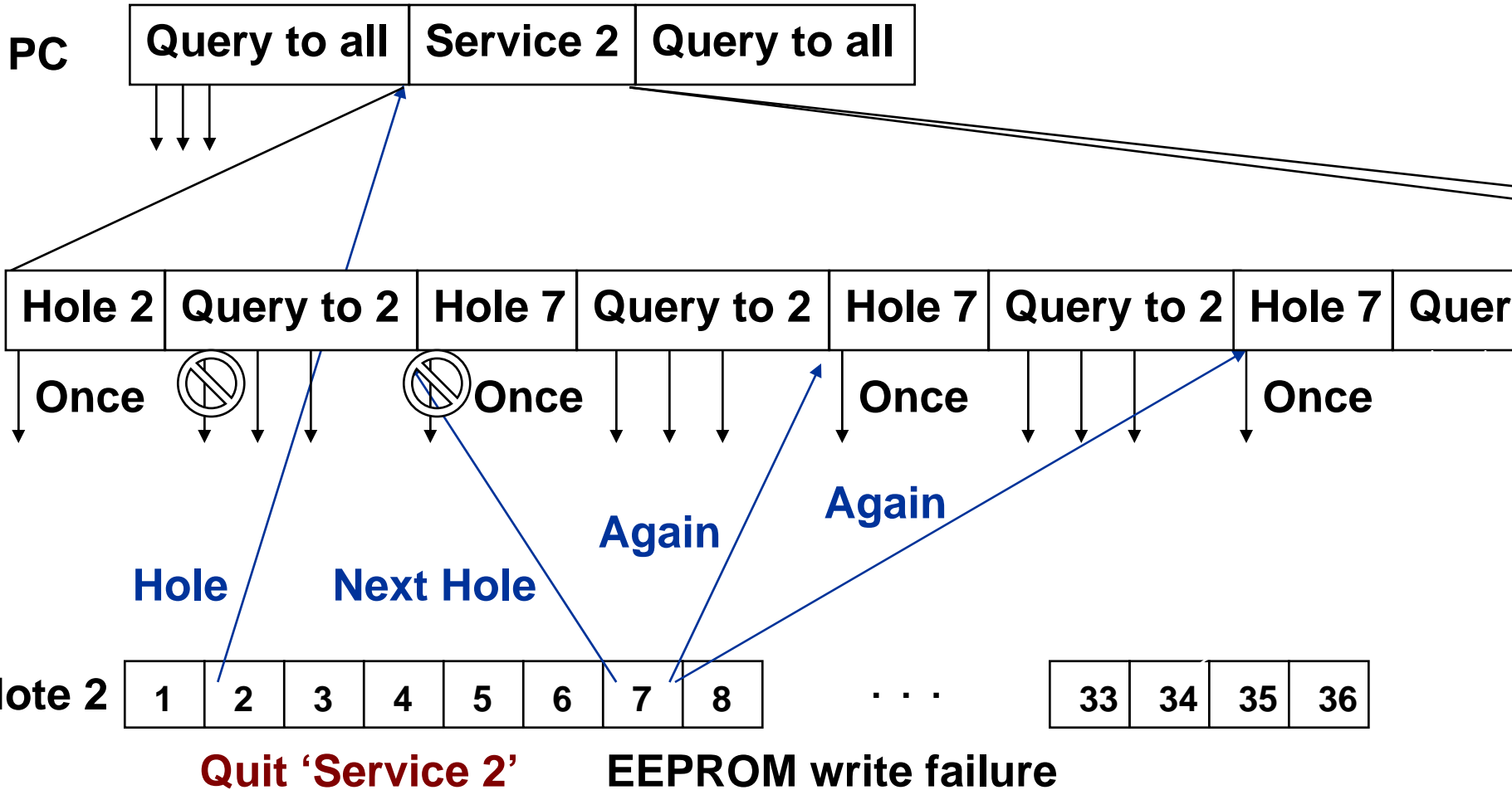


Query Phase **Timeout** *Failure !!!*



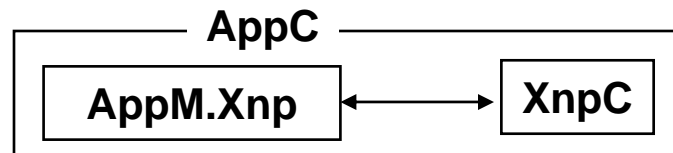
Eliminate timeout using positive ack Modification (continued)

Query Phase

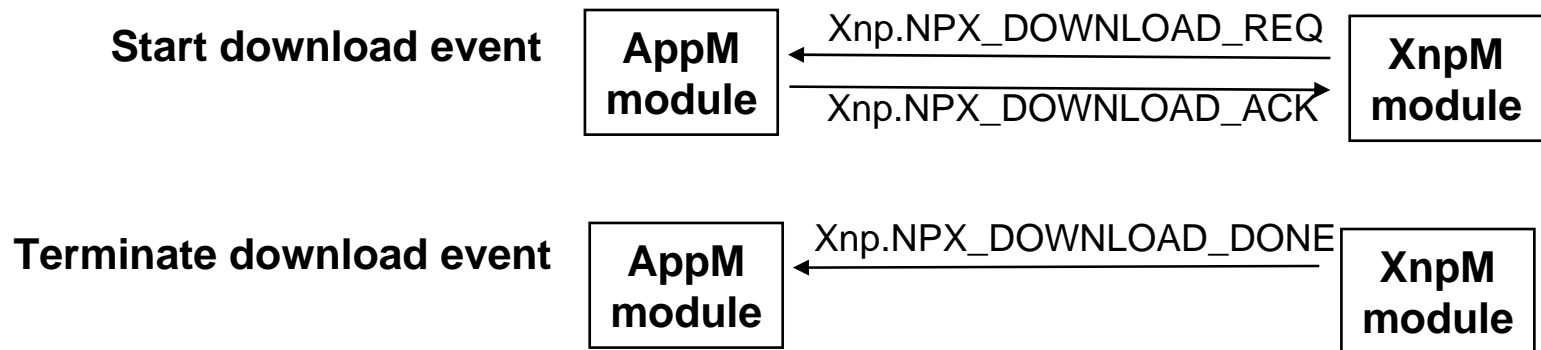


How to wire apps for network programming

- In configuration file (AppC), XnpC module should be connected to the implementation (AppM).



- In AppM, two events should be handled.



Location of Network programming module

- **Check out these directories in CVS.**
 - **Tinyos-1.x/apps**
 - » **XnpService: main module for network programming.**
 - » **XnpCount: a sample app**
 - » **XnpRfmToLeds: a sample app**
 - **Tinyos-1.x/tools/java/net/tinyos**
 - » **Util_xbow: a modification to SerialStub**
 - » **Xnp: xnp java program**

Installing network programmable code

- A network programmable code should be loaded using parallel programming in the beginning.
1. Compile an application in ATmega128 native.
 - Currently, supported in mica2, mica2dot.
 - Latest tool: avr-gcc, nesC 1.1 and uisp needed.
 - Refer to <http://webs.cs.berkeley.edu/tos/mica2.html>

2. Load app binary in ATmega128 native mode.

```
uisp -dprog=dapa --wr_fuse_e=ff --erase
```

```
uisp -dprog=dapa --wr_fuse_e=ff --upload if=main.srec
```

```
uisp -dprog=dapa --wr_fuse_e=ff --verify if=main.srec
```

3. Load boot loader.

```
uisp -dprog=dapa --upload if=bootloader.srec
```

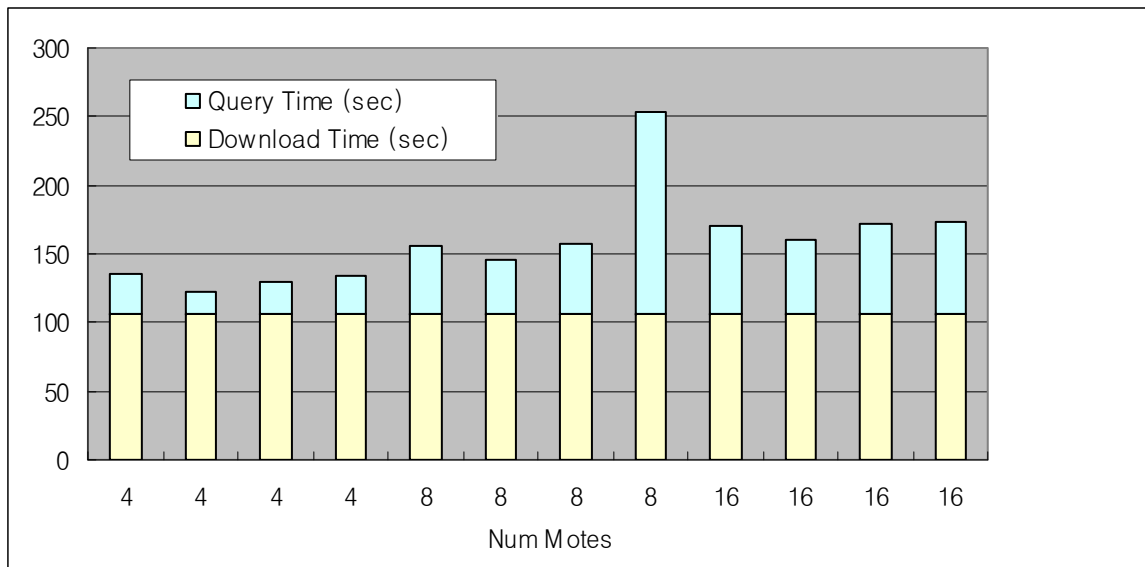
Issues in PC side.

- **Xnp java program**
 - sends a command, a program capsule and hears the response from the motes.
 - sends/receives radio packets through UART.
- **Different versions of GenericBase.**
 - **GenericBase**
 - » Easy to lose sync. Not recommended for network programming.
 - **Sync GenericBase**
 - » Check sync bytes of UART packet. Works for both direct access and SerialForward
 - **XGenericBase**
 - » Check sync bytes of UART packet. Only works for direct access
- **Sync GenericBase with SerialForward is recommended.**

Test Results

- **Test program: XnpCount**
 - **Program size: 37000 bytes, 841 capsules.**

Num Motes	Num Tries	Download Time (s)	Query Time (s)	Num Successes
4	4	106	24.25	4
8	4	106	72.25	7.75
16	4	106	63	16



Test Results (Cont.)

- **We could get reasonable success rate when motes are normal.**
- **Cases when network programming fails.**
 - **Parameters in xnp java is not correct.**
 - » **When xnp sends packets faster than GenericBase can handle.**
 - » **This can halt network programming globally.**
 - » **Interval between each write should be large enough for GenericBase to handle. 120 ms was found by experiment.**
 - **Program code is not correctly loaded initially.**
 - » **Mote doesn't go into download mote.**
 - **A mote cannot write into its EEPROM.**
 - » **This happened when battery voltage was low. And was fixed with new battery.**
 - **After hardware modification was made.**
 - » **A mote didn't work properly when we solder the loose battery connection.**
 - » **We needed to load the code with parallel programming.**

Discussion

- **Can we assume that program image is correct just by checking PID and CID are correct?**
- **When mote #1 loses packet K, does its neighbor mote #2 also lose the packet?**
- **Experiments showed that query time increases with the number of motes. Will it be worthwhile to use Forward Error Correction (FEC) for larger number of motes?**
- **Does network programming justify the higher power consumption in favor of convenience?**
- **May need characteristic numbers about motes and TinyOS**
 - Radio Speed, Packet loss rate
 - EEPROM Speed (read, write), write failure rate
 - Atomicity of EEPROM write