

Hands-On Tutorial with OpenThread

This lab procedure walks through the steps to create a Thread network and an OpenThread Border Router. The first part reviews how to create an OpenThread project in Simplicity Studio v5. The second part shows how to create a network with three nodes, and how to use some basic commands to explore the feature of each node. The final part introduces how to remove and add a node, and analyze the status of each node. A demonstration of commissioning ends this session.

KEY POINTS

- Create OpenThread project for EFR32MG12
- Create a Thread network
- Add and remove a node and analyze the network topology change
- Create an OpenThread Border Router
- Use commissioning with Thread
- Use Silicon Labs tools to analyze the network

1 Prerequisites

For this lab you will need the following:

- Three EFR32MG12 radio boards this tutorial assumes BRD4170A, although other EFR32MG12 radio boards would work too with possible minor adaptation in instructions
- Three Mini-USB Type-B to USB Type-A cable supplied with any Wireless Gecko starter kit
- Simplicity Studio v5
 - o GNU ARM V7
 - Gecko SDK Suite 3.1.0 or later
- (Optional) a serial terminal such as Tera Term

2 Create an OpenThread Project

We want to create three OpenThread projects and then create a network:

- ✓ child project
- ✓ router_eligible_1 project
- ✓ router_eligible_2 project
- 1. Launch Simplicity Studio from your desktop
- 2. Connect the 4170A radio board with EFR32MG12 to your PC using the USB cable
- 3. When the device is connected to your PC, you should see it listed in the Debug Adapters window in Simplicity Studio



4. File >> New >> Silicon Labs Project Wizard

Si	v5_workspace - MyLight/zcl-framework-core/af-main-soc.c - Simp	licity Studio™	
Fil	e Edit Source Refactor Navigate Search Project Run Windo	w Help	
	New	Alt+Shift+N >	Silicon Labs Project Wizard
	 Open File (Network Analyzer Trace, Energy Profiler, etc.) Open Recent File (Network Analyzer Trace, Energy Profiler, etc.) Other Network Analyzer Actions Recent Files 	>	Project. Create a C/C++//Assembly project targeting Silicon Labs MCUs. Dither Difficity Studio
	Close	Ctrl+W	velop, research, and configure devices for IoT applications.
	Close All	Ctrl+Shift+W	
	Save	Ctrl+S	
6	. Save As		
	Save All	Ctrl+Shift+S	
	Revert		or search for a product by name to see available documentation,
	Move		
	Rename	F2	
	Refresh	F5	
	Convert Line Delimiters Io	>	ducts
	Print	Ctrl+P	
2	Import Export		
	Switch Workspace		
-	Network Analyzer Import		
	Properties	Alt+Enter	
	Restart		
_	Exit		

- 5. If your radio board and WSTK are detected automatically, a new window shows up
- 6. Make sure you use Gecko SDK 3.1.0 or later and GNU ARM v7.X.X compiler
- 7. Click on NEXT

New Project Wizard							
Target, SDK, and Toolchain Sel	ection						
Select the target board, device, SDK, and IDE/toolchain to use for the project.							
🕐 Target, SDK	Examples	Configuration					
larget Boards:							
Search or Select		•					
EFR32MG12 2400/868-915 MHz 19	dBm Dual Band Radio Board (BRD4170A Rev A00) 😒						
Terret Device							
Target Device:							
Search or Select		·					
EFR32MG12P433F1024GM68							
SDK:							
Select SDK							
Gecko SDK Suite: Bluetooth 3.0.0.2, B	mberZNet 6.8.0.1, Flex 3.0.0.2, Micrium OS Kernel, Op	penThread 1.0.0.2 (GitHub-f411a412bee),					
		📩 Manage SDKs					
IDE / Toolchain:							
Colort IDE / Taolohain		.					
Select IDE / Toolchain							
Simplicity IDE / GNU ARM v7.2.1							
Simplicity IDE / GNU ARM v7.2.1							
Simplicity IDE / GNU ARM v7.2.1							

2.1 Minimal Thread Device project

The node **child** project is created with *ot-cli-mtd* example. The child is linked with a parent device (leader, router). In a network the child can act in two different ways:

-As a Sleepy End Device (SED), in the software, we need to wake it up to check if there is a new message

-As a Minimal End Device (MED), its radio device is always on, we do not need to poll to have an update.

In our case, we will use it as a MED.

- 8. Now, we can tick the thread check box,
- 9. Select the ot-cli-mtd example only for the first node and click on next. The other two nodes use ot-cli-ftd example
- 10. Then click on NEXT



✓ Target, SDK ───	Configurat				
Filter on keywords	8 resources found				
Technology Type S Clea	r Filter Ot-ncp-ftd This is a simple OpenThread Full Thread Device NCR application				
Bluetooth (10)					
Bootloader (9)	ot-oli mtd				
Platform (6)	This is a very simple CLI application to test the Openthread stack on a				
Proprietary (19)	Minimal Thread Device.				
🗹 Thread (8)					
Zigbee (14)	ot-cli-ftd This is a very simple CLI application to test the Openthread stack on a Full				
	Thread Device.				
	ot-ble-dmp This is a simple application to test DMP with OpenThread and Bluetooth				

- 11. For ot-cli-mtd, enter the name "child".
- 12. Click on FINISH

lew Project Wizard		— 🗆 X
roject Configuration		
elect the project name and locati	on.	
Target, SDK	Examples	Configuration
Project name: child		
✓ Use default location		
Location: C:\Users\yanecib\Simpl	icityStudio\v5.rel.Staging_388\child	BROWSE
With project files:		
 Link to sources 		
Link sdk and copy project sour	rces	
O Copy contents		

2.2 Full Thread Device Project

The **router_eligible_1** project is created with *ot-cli-ftd* example. The **router_eligible_1** will be the *leader* when the network first forms. The **router_eligible_1** can act as a router, Router Eligible End Device, or Full End Device.



- 13. Repeat step 4 to 8 Create an OpenThread Project to create two projects, name the first one router_eligible_1 and the other one router_eligible_2
- 14. Select the *ot-cli-ftd* example (both projects) and click on **NEXT**

xample Project Selection						
elect the project template to open in Sim	plicity IDE.					
V Target, SDK	Configuration					
Filter on keywords	8 resources found					
Technology Type S Clear Filter	ot-ncp-ftd This is a simple OpenThread Full Thread Device NCP application					
Bluetooth (10) Bootloader (9) Platform (6) Proprietary (19)	ot-cli-mtd This is a very simple CLI application to test the Openthread stack on a Minimal Thread Device.					
 Thread (8) Zigbee (14) 	ot-cli-ftd This is a very simple CLI application to test the Openthread stack on a Full Thread Device.					
CANCEL	ot-ble-dmp This is a simple application to test DMP with OpenThread and Bluetooth BACK NEXT FINISI					

3 Compile and Flash Device Firmware

15. Select your child project and click on the hammer to compile your project. Repeat this step for the router_eligible_1 and router_eligible_2



File Edit Navigate Search Project Run Window Help

🏘 🕶 💁 🕶 📄 🐨 🔚 🕲 💌 🔦 💌 💷 🏦 Welcome 🕑 Re	Recent ⅲTools 圡Install 幸 Preferences							
🔁 Project Explorer 🛛 🗧 😫 🏹 🕴 🗖 🗎	1							
Y 😂 child [GNU ARM v7.2.1 - Debug] [EFR32MG12P332F1	1							
> 🔊 Includes								
> 🗁 autogen								
> 🗁 config								
> 🗁 gecko_sdk_3.1.0								
> 😕 GNU ARM v7.2.1 - Debug								
> 🖻 app.c								
> 🖻 app.h	Building project							
> 🖻 main.c								
🐣 child.slcp								
🙆 ot-cli-mtd.pintool								
ot-cli-mtd.slps	Invoking Command: make -j8 all							
> 📽 router_eligible_1 [GNU ARM v7.2.1 - Debug] [EFR32M	Λ							
> Sources of the second sec								
	Always run in background							
	Run in Background Cancel Details >>							
< >								
🗬 Debug Adapters: 3 🛛 📴 Outline 🛛 🖓 🗖								
\$ X \ 2 X X ♥ ▼ ■ 🗎 🕀 🕀	Problems							
	CDT Build Console [child]							
	arm-none-eabi-g++ -g3 -gdwarf-2 -mcpu=cortex-m4 -mthumb -std=c++0x '-DRTT_USE_ASM=0' '-D	OPEN						
> 🖗 EFR32MG12 2.4 GHz 19 dBm RB (ID:440176275)								

16. Wait a few minutes until the full project is built. Once it is successfully finished, you can read this message in the Console tab:



4 Node Debug setup

4.1 Rename Your Board

17. Rename your device to display each device role In **Debug Adapter review**, right-click on your device and select **Rename**, then a "Rename Device" window opens.

🚚 Debug Adapters: 2 💈	🛛 📴 Outline
* 💥 🗳 [
> \$ EFR32MG12	Cup to Jp., DR (ID. 44041
> • EFR32MG12	Disconnect
	Start capture
	Start capture with options
	Stop capture
	Redo last upload
	Upload application
	Upload adapter firmware
	Rename
	Make a sniffer
<u>100</u>	Launch Console Spiffer Configurator
	Davies configuration.
8	Earce Unlock
Debug Adapters: 3 🖾 🗄 Outline	3 Rename Device
\$ 🗙 🖬 🖉 🛪 🛠 🌩 🗖 🖨 🖻	
>	Choose a new name for this device
	child
> \(\phi \) EFR32MG12 2.4 GHZ 19 dBm RB (ID:440176275)	
	17
1	OK Cancel

18. For each project, to flash the corresponding .hex file, expand the **Binaries folder** under your project, right-click on the .hex, and select **Flash to Device...**

Project Explorer 🛛		🖻 😫 🎖 🕴 🗖 🛙									
Y 🖆 child [GNU ARM	∕l v7.	2.1 - Debug] [EFR32MG12P332	~								
👻 🖑 Binaries	Y 🖑 Binaries										
> 🕸 child.axf	- [arr	m/le]									
> 🖸 child.bin	- [un	known/le]									
> O child.hex	- ſur	known/le1									
> O child.s37		New									
> 🔊 Includes		Open									
> 🗁 autogen		Show In	Alt+Shift+W >								
> conig		Open With	>								
> GNU ARM v		Сору	Ctrl+C								
	Ē	Paste	Ctrl+V								
> 🖻 app.h	×	Delete	Delete								
> 🖻 main.c		Move									
🚢 child.slcp		Rename	F2								
<		Import	>								
Debug Adapters: 3		Build Project									
> © FER32MG12.2	8	Refresh	F5								
>	0	Run As	>								
> # EFR32MG12 2.	*	Debug As	>								
		Profile As	>								
		Team	>								
		Compare With	>								
		Replace With	>								
		Browse Files Here									
		Open Command Line Here	N								
		Flash to Device	2								
		Properties	Alt+Enter								

19. Select your device in the window and click OK

∲ child · FER3	2MG12P332F102	4GI 125 (II	0.44017630	
router_eligi	ble_1 : EFR32MG1	12P432F10	24GL125 (I	
router_eligi	ble_2 : EFR32MG1	12P332F10	24GL125 (I	
Romombor	my decision			_

- 20. Click on Advanced Settings
- 21. Advanced settings window opens, click on Full Erase and OK
- 22. Then on the Flash Programmer window click on Program
- 23. A **Programming Flash** window will be opened, the firmware is being flashed.

Device			
Board Name: Wireless Starte	er Kit Mainboard	/	•
Board Name: EFR32MG12 2			Ŀ
MCU Name: EFR32MG12P3	Advanced Settings	×	
Adapter	Flash Erase Options		
Name: child	O Merge Content		
	Page Erase A s u s	21	
Flash Part	Full Erase		
File Type hex	? ОК	Cancel	
File			
ıdio\v5.rel.Staging_388\chi	ild\GNU ARM v7.2.1 - Debug	g\child.hex ~ Browse	
Advanced Settings 20		Frase Program	
		Liase	
Flash Erase/Write Protecti	on		
Select flash range	✓ 0x0	✓ 0x100000	
○ Select default sections	🗹 Lock Main Flash	Lock User Page	
	Protect	Remove Protection	
Debug Lock Tools			
The unlock function only we Unlocking the chip will erase	orks using Silicon Labs EFM3 e all data on flash and SRAM	2 and EFR32 boards. I.	
	Unlock Debug Acce	ess Lock Debug Access	
			/
		cl	
Ø		Close	
Search Flash Programmer		— 🗆 🗙	
Change Device			
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F	rter Kit Mainboard : 2.4 GHz 10 dBm Radio Boa ?332F1024GL125	rd 22	
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125	rd 22	
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125	rd 22	
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter ramming Flash	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125	rd 22	
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter ramming Flash Copying C\Users\yanecib\Sir	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter ramming Flash Copying C:\Users\yanecib\Sir	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 (************************************	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter ramming Flash Copying C:\Users\yanecib\Sir	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 3332F1024GL125 mplicityStudio\traichild.he:	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 3332F1024GL125 mplicityStudio\traichild.he 00011FFF (2 KB)	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12F Adapter ramming Flash Copying C:\Users\yanecib\Sir	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB)	rd 22 – x to J-Link Silicon Labs (44017/	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.her 00011FFF (2 KB)	rd 22 – O	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter amming Flash Copying C:\Users\yanecib\Sit	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background	rd 22 	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x iys run in background	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12I MCU Name: EFR32MG12I Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x iys run in background	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x ays run in background	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background V 0x0 ~ -	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12I Adapter Adapter Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x ays run in background	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background V 0x0 ~ - V 10x0 ~ -	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x ays run in background	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background V 0x0 A - V 10x0 A - V 10x0 A -	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x ays run in background Select flash range Select default sections Debug Lock Tools The unlock function only v	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background V 0x0 A - V 0x0 A - V 10ck Main Flash Protect vorks using Silicon Labs EFM	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sir nming range 0x00011800 - 0x nys run in background Select flash range Select default sections Debug Lock Tools The unlock function only v Unlocking the chip will era	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background Vox0 A - Vox0 A - Voxks using Silicon Labs EFM se all data on flash and Sama	rd 22	627
Device Board Name: Wireless Sta Board Name: EFR32MG12 MCU Name: EFR32MG12 Adapter ramming Flash Copying C:\Users\yanecib\Sin nming range 0x00011800 - 0x ys run in background Select flash range Select default sections Debug Lock Tools The unlock function only v Unlocking the chip will era	rter Kit Mainboard 2.4 GHz 10 dBm Radio Boa 2332F1024GL125 mplicityStudio\traichild.he: 00011FFF (2 KB) Run in Background V 0x0 A - Vorks using Silicon Labs EFM se all data on flash and SRA Unlock Debug Ac	rd 22	627

4.2 Open Console to Communicate with Your Board

24. Open a console for each project in the **Debug Adapter** window: Right-click on the device and select **Launch Console...**

Fil	e	Edit	Navi	gate	Sea	rch	Project	Run	Windo	w	Help		
*	•	8	- 1 📬	-		8	- % -	1	Welco	me	🕑 Re	cent	
•	Pro	oject	Explo	rer 🛛				E	3 😫 🍸	000			
>	B	chilo	I [GNI	J ARI	M v7	.2.1	- Debug] [EFR	32MG1	2P3	32F1		
>	ß	rout	er_eli	gible_	1 [G	NU .	ARM v7.	2.1 - E	ebug] [[EFF	R32M		
>	ß	rout	er_eli	gible_	2 [G	NU .	ARM v7.	2.1 - E	ebug]	[EFF	R32M		
<											>		
₽ł:	De	bug	Adap	ters: 8	3 🛙	₽ C	Outline						
			1.00			% ¥	(🗋 🖉)	< 💥 🕻	> 🗖 🔲	P	⊟ ⊞		
Ľ	Ŷ	chile	d (ID:4	44017	/630	5)	Ronam	0					
Ľ	Ψ	rou	ter_eli	gible_	_1 (IL _2 //F		Conno	re ct					
Ľ	Ψ	rou	ter_ell	gible_	_2 (IL		Discon	nact			6		
							Chart	inect			Ľ	7	
							Start G	apture					
							Start ca	apture	with op	otio	ns		
							Stop c	apture					
							Redo l	ast up	load				
							Upload	d appli	ication	•			
							Upload	d adap	ter firm	wa	re		
							Make a	a sniffe	er				
						>-	Launch	Cons	ole				
						T(()	Sniffer	Confi	gurator.				43
						•	Blueto	oth NO	CP Com	ma	nder		
						*	Device	config	guratior	n			
						2	Force l	Jnlock					

25. Placing the two console windows side by side, as shown in the picture below, can make it easier to switch your active console window from one node to another. We use **serial 1** to communicate with the device.

File Edit Navigate Search Project Run Window Help		
🎋 🕶 🥵 🕶 📑 🕶 🔛 💿 💿 🖛 🍕 💌 🕮 🖍 Welcome 💿 Re	ecent ⊞ Tools ≛ Install ≉ Preferences	😫 🛿 Launcher 🚺 Simplicity IDE
Project Explorer ≅	😢 child 🛛	🖉 router eligible 1 🖂
> 🖆 child [GNU ARM v7.2.1 - Debug] [EFR32MG12P332F1		
Fouter_eligible_1 [GNU ARM v7.2.1 - Debug] [EFR32M]	Ine terminator: CR-LF (DOS, C	No translation V Line terminator: CR-LF (DOS, C
> Source_ligible_2 [GNU ARM v7.2.1 - Debug] [EFR32M	🚔 Serial 0 🚔 Serial 1 🚔 Admin 🚔 Debug	🚔 Serial 0 🚔 Serial 1 🚔 Admin 🚔 Debug
	^	^
< >		
🕷 Debug Adapters: 3 🛛 🏂 Outline 👘 🗖		
% 💥 🖸 🖉 💥 🌣 ▼ 🗐 🖼 🕀		25
> thild (ID:440176305)		
v router_eligible_1 (ID:440176275)		_
	×	×
	×	×

26. From the top bar menu, click on window and select Preferences >> Network Analyzer >> Decoding >> Stack Versions

	Window Help Network New Window Editor > Appearance > Show View > Perspective > Preferences	
Si Preferences		— 🗆 X
type filter text	Stack Versions	⇔ ▼ ⇔ ▼ §
 > C/C++ > Help > Install/Update ✓ Network Analyzer Capture Configuration Capture File Storage Connectivity Display ✓ Decoding Bluetooth Frames and Fields Reports Security Keys Stack Versions Transaction Groupers Energy Profiler Integration Node Icons 	Z-Wave Auto-detecting decoder stack Bluetooth Low Energy Custom Stack Connect stack Ember RF4CE stack ØpenThread stack EmberZNet 6.8 EmberZNet 6.7 EmberZNet 6.6 EmberZNet 6.5 EmberZNet 6.4 EmberZNet 6.3 EmberZNet 6.2 EmberZNet 6.1	26
? è 🗹		Apply and Close Cancel

5 Lab1: Create a Thread network

5.1 Create the Network

To create the network, we start with the **router_eligible_1**. From its console enter the commands lines below:

Router_eligible_1 console		
Command-line	Expected Response	Description
> dataset init new	Done	Create a new network configuration
> dataset commit active	Done	Commit new dataset to the Active Opera- tional Dataset in non-volatile storage.
> ifconfig up	Done	Enable Thread interface
> thread start	Done	Enable and attach Thread protocol opera- tion.
	Wait about 10 seconds	
> state	leader	Read the current status : offline, disa- bled, detached, child, router, or leader
> dataset	<pre>> dataset Active Timestamp: 1 Channel: 15 Channel Mask: 0x07fff800 Ext PAN ID: fa8f8d5f1cd2b5e9 Mesh Local Prefix: fd6e:c30c:24e9:e3d9::/64 Master Key: 97e05cb1f2df6f0d9af3026448f4b834 Network Name: OpenThread-2136 PAN ID: 0x2136 PSKc: 47c8cd2bed340bab86c47929cf0e99db Security Policy: 0, onrcb Done</pre>	View network configuration

According to the response of the command state and dataset, the leader is created.

5.2 Add a Node to Our Network

To add a node to our network, enter the commands lines below in the **child** console:

Child console				
Command-line	Expected Re- sponse	Description		
> dataset channel <mark>15</mark>	Done	To decrease latency, the node needs to use the channel used in the alive network.		
<pre>> dataset masterkey 97e05cb1f2df6f0d9af3026448f4b834</pre>	Done	Only the Master Key is required for a device to attach to a Thread network		
> dataset commit active	Done	Commit new dataset to the Active Operational Dataset in non-vola- tile storage.		

> ifconfig up	Done		
		Enable Thread Interface	
> thread start	Done		
		Enable and attach the Thread protocol operation.	
	Wait about	20 seconds	
> state	child		
		Read the current status: offline, disabled, detached, child,	
		router or leader	

<pre> # router_eligible_1 </pre>	literation of the second seco
No translation	No translation
≧ Serial 0 ≧ Serial 1 ≧ Admin ≧ Debug	🛎 Serial 0 🚔 Serial 1 🚔 Admin 🚔 Debug
dataset init new A	dataset channel 15 Done
> > dataset commit active Done	> > dataset masterkey 97e05cb1f2df6f0d9af3026448f4b834 Done
> ifconfig up Done	> dataset commit active Done
> > thread start Done	> > ifconfig up Done
<pre>> dataset Active Timestamp: 1 Channel: 15 Channel: 15 Channel Mask: 0x07fff800 Ext PAN ID: fa8f8d5f1cd2b5e9 Mesh Local Prefix: fd6e:c30c:24e9:e3d9::/64 Master Key: 97e05cb1f2df6f0d9af3026448f4b834 Network Name: OpenThread-2136 PAN ID: 0x2136 PSKc: 47c8cd2bed340bab86c47929cf0e99db Security Policy: 0, onrcb Done > state leader Done > > state</pre>	> thread start Done > state child Done > >

5.3 Explore Deeper in Our Network

After our network is created, we want to check the environment and use the Network Analyzer in Simplicity Studio

1. Enter the security key: Window >> Preferences

type filter text	Security Keys			⇔ ▼ ⇔ ▼
General	Check to activate. Enter 16 char	ASCII or 32 char Hex		
> Help	A T Name	Кеу	Touched	New
Install/Update	🗹 [App Bootloader Link	65 6D 62 65 72 20 45 4D 32 35 30 20 61 62 6C 4C	Nov 30	Import
> MCU	🗹 [App Bootloader Network	65 6D 62 65 72 20 45 4D 32 35 30 20 61 62 6C 4E	Nov 30	import
Network Analyzer	🔽 [ConnectSampleAppKey	AA	Nov 30	Clone
Capture Config	☑ [Default Framework V1 Ln	05 05 05 05 05 05 05 05 05 05 05 05 05 0	Nov 30	Delete
Capture File St	☑ [Default Framework V1 N	04 04 04 04 04 04 04 04 04 04 04 04 04 0	Nov 30	
Connectivity D	🗹 [Default IP Test Event Key	BF BE BD BC BB BA B9 B8 B7 B6 B5 B4 B3 B2 B1 B0	Nov 30	Invert
✓ Decoding	☑ [Default OWL AES Key 1	12 34 56 78 9A BC DE F0 12 34 56 78 9A BC DE F0	Nov 30	Clear All
Bluetooth	☑ [Default OWL AES Key 2	22 22 22 22 22 22 22 22 22 22 22 22 22	Nov 30	Run HMAC.
Frames and	✓ N ThreadTraining	97 E0 5C B1 F2 DF 6F 0D 9A F3 02 64 48 F4 B8 34	Today	
Reports	☑ [Sensor/Sink Link	5A 69 67 62 65 65 20 53 65 63 75 72 69 74 79 21	Nov 30	ASCII edit
Security Key	☑ [Sensor/Sink Network	65 6D 62 65 72 20 45 4D 32 35 30 20 63 68 69 70	Nov 30	
Stack Versic	☑ [Smart Energy Link	12 33 33 33 33 33 33 33 33 33 33 33 33 33	Nov 30	
Transaction	☑ [Smart Energy Network	56 77 77 77 77 77 77 77 77 77 77 77 77 77	Nov 30	
Energy Profiler	☑ [Standalone Bootloader Li	65 6D 62 65 72 20 45 4D 32 35 30 20 6C 69 6E 6B	Nov 30	
Node Icons	☑ [Standalone Bootloader N	65 6D 62 65 72 20 45 4D 32 35 30 20 6E 77 6B 20	Nov 30	
Optional Dialo	☑ [Super Parent Link	53 75 70 65 72 50 61 72 65 6E 74 4C 69 6E 6B 4B	Nov 30	
Stream Visuali:	☑ [Super Parent Network	53 75 70 65 72 50 61 72 65 6E 74 4E 65 74 77 4B	Nov 30	
Timeline	☑ [ZigBeeAlliance09	5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39	Nov 30	
Wireshark				
> Run/Debug				
Simplicity Studio	1			
> leam				
Ierminai	Save decryption keys in ISD file	es ∠ Disable keys when not used for 365 days		
Vireless Develop			Restore Defaults	Apply
?			Apply and Close	Cancel

2. In the Debug adapter, right-click on router_eligible_1 and select Start capture, repeat this step for the child.



3. Click on Network Analyzer button to switch the view

🧕 training_workspace - ot-rcp/ot-rcp.slcp - Simplicity Studio™		
<u>File E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp		
	😰 🖉 Launcher 🚯 Simplicity IDE	🕹 Network Analyzer

The network analyzer opens, feel free to set up your consoles as below to have a panoramic view.

-Highlight a network transaction such as the "MLE Child Update" shown in the example below, and in the "Event Detail" window you can find the PAN ID of your network. In my example, the PAN ID is 0x2136. Yours may be different.

	⊾*Live ≋										
Γ	2 saved filters	AND									✓ ≱ © ©
	.00 n/s	34.993	1								,
P	0.000s. M.		nath an an an at a	A					<u> </u>		
	Time:34.992903	s Real ti	me:N/A Nodes:2 Event:UD	P Message							💐 Event Detail 🗢 🗆
Г			<u> </u>								MLE crypto: ROOT, 97 E0 5C B1 F2 DF 6F 0D 9A F3 02 64 🔨
L	•		 •								✓ IEEE 802.15.4 [22 bytes]
L	000440176305 child		0004401	76275							PHY Header: 0x52
L	F002		F000								Packet Length: 82
L											Frame Control: 0xDC61
L											Frame Type: Data (1)
L											Security Enabled: false
L											Frame Pending: false
L											Ack Required: true
L											Intra Pan: true
L											Frame Version: 2006 (1)
L	Transactions	otal:22 sh	own:22							~ 🗆	Reserved: 0x00
Г	Time	Durati	Summary	IPv6 Src	IPv6 Dest	P#	M#	E#	Error Status	Warning ^	Destination Address Mode: Long (3)
j,	34.992903	0.003	MLE Child Update	fe80::8065:91	fe80::a447:e8	2					Source Address Mode: Long (3)
Г	34.997364	0.005	MLE Child Update Respon	fe80::a447:e8	fe80::8065:91	2					Sequence: 0xE7
L	35.992260	0.003	MLE Child Update	fe80::8065:91	fe80::a447:e8	2					Destination PAN ID: 0x2136
L	35.996727	0.005	MLE Child Update Respon	fe80::a447:e8	fe80::8065:91	2					Long Destination Address: A647E8537EA50834
L	36.993674	0.003	MLE Child Update	fe80::8065:91	fe80::a447:e8	2					Long Source Address: 826591EE3CDA032E
L	36.997175	0.010	MLE Child Update Respon	fe80::a447:e8	fe80::8065:91	3	1				> 6Lowpan [2 bytes]
L	37.993232	0.003	MLE Child Update	fe80::8065:91	fe80::a447:e8	2					✓ Lowpan UDP [7 bytes]
L	37,997054	0.005	MLE Child Update Respon	fe80::a447:e8	fe80::8065:91	2					Next Header Encoding: 0xF0
L	39.579151	0.006	MLE Parent Response	fe80::a447:e8	fe80::8065:91	2					NHC ID: UDP (30)
L	40.057114	0.004	MLE Child ID Request	fe80::8065:91	fe80::a447:e8	2					Checksum Elided: false
	40.064889	0.003	MLE Child ID Response	fe80::a447:e8	fe80::8065:91	2				<u> </u>	Port Mask: Full Ports (0)
b	<	51000	the child to the points			-				>	Source Port: 0x4D4C
F										-	Dest Port: 0x4D4C

You can also rename your device represented in the upper view: right-click on it and choose label, then enter the desired name



5.4 A Quick Look at Our IP Addresses

Router_eligible_1 console		
Command line	Response	Description
> ipaddr	<pre>> ipaddr fd6e:c30c:24e9:e3d9:0:ff:fe00:fc00 fd6e:c30c:24e9:e3d9:0:ff:fe00:f000 fd6e:c30c:24e9:e3d9:fb6d:7770:b132:5823 fe80:0:0:0:a447:e853:7ea5:834 Done</pre>	Display the ipv6

Child console		
Command line	Response	Description
> ipaddr	> ipaddr	Display the ipv6

fd6e:c30c:24e9:e3d9:0:ff:fe00:f001	
fd6e:c30c:24e9:e3d9:6fa:ecc5:34b1:27c5	
fe80:0:0:0:8065:91ee:3cda:32e	
Done	

5.4.1 Link-Local addresses (LLA)

This address starts with fe80::/16 prefix (for instance fe80:0:0:a447:e853:7ea5:834), it is created with the MAC address. It is not used to communicate between nodes. We can still use them between two nodes if there is only a link, one radio transmission, not more than one cable to retransmit the message.

5.4.2 Example of building LLA (router_eligible_1):

Mac address:	A647:e853:7ea5:834
Flip the seventh bit	A6 = 1010 01 <mark>1</mark> 0 A4 = 1010 01 <mark>0</mark> 0
LLA	FE80::A447:e853:7ea5:834

5.4.3 Routing Locator Address (RLOC)

fd6e:c30c:24e9:e3d9:0:ff:fe00:f000 (Router_eligible_1), this address is created when the device is attached to the network, and is generally not used by applications. The blue part is the mesh prefix. This address changes if the topology changes, in other words, if you remove/add a device.

5.4.4 Mesh Local Address (ML-EID)

fd6e:c30c:24e9:e3d9:fb6d:7770:b132:5823 (Router_eligible_1), this address is independent of the network topology, and is used to communicate with the other interface in the same thread network.

5.4.5 Anycast (only the leader)

fd6e:c30c:24e9:e3d9:0:ff:fe00:fc00 (router_eligible_1), this is anycast address, it is used to route traffic to a Thread interface when the RLOC of a destination is not known. An Anycast Locator (ALOC) identifies the location of multiple interfaces within a Thread partition. The last 16 bits of an ALOC, called the ALOC16, is in the format of 0xfcXX, which represents the type of ALOC.

ALOC16	Туре
0xfc00	Leader

To sum up, only the leader has an anycast address, the LLA is built from the MAC address, the ML-EID is created when the network is up and does not change, the RLOC changes if the topology of the network changes as well. To communicate between the interface of the same mesh network, we use the ML-EID. To identify an interface we use RLOC. Here is a link to study deeper IPV6 addressing: https://openthread.io/guides/thread-primer/ipv6-addressing

Nature	Communication	Description	Comments
LLA	АВ	Yes, both direction	Point to point
	AC	A to B yes, both direction B to C yes, both direction A to C no, any direction	Only one radio transmission, useful to discover the neighbor or routing.
RLOC		A to B yes, both direction B to C yes, both direction A to C yes, any direction	If you change the topology, the ad- dress change.
ML-EID			Any interface of the same mesh net- work, so here my address which starts with fd6e:c30c:24e9:e3d9

5.4.6 Zoom in on the "child table" command line

	Router_eligible_1 console									
Command-line	Example of an Expected Response	Description								
> child table	ID RLOC16 Timeout Age LQ In C_VN R S D N Extended MAC	Attached child info								
	1 0xf001 240 38 3 47 1 1 0 0 826591ee3cda032e									

5.5 Understanding Leader, Router, and Child Roles

We want to add a device and then remove the current leader to understand some of the healing properties of Thread networks.

Launch the console of the router_eligible_2 and repeat command lines in the section Add a Node to Our Network.

Ørouter_eligible_1 ≅	<pre> # router_eligible_2 </pre>	ể child ¤
No translation	No translation	No translation
🛎 Serial 0 🚔 Serial 1 🎽 Admin 🚔 Debug	Serial 0 Serial 1 Admin Debug	🛎 Serial 0 🚔 Serial 1 🚔 Admin 🚔 Debug
<pre>state leader Done > child table ID RLOC16 Timeout Age LQ In C_VN R D N Extended MAC +++ +++ 1 0xf001 240 23 3 72 1 0 0 826591ee3cda032e Done > ipaddr fd6e:c30c:24e9:e3d9:0:ff:fe00:fc00 fd6e:c30c:24e9:e3d9:0:ff:fe00:fc00 fd6e:c30c:24e9:e3d9:0:ff:fe00:f000 fd6e:c30c:24e9:e3d9:fb6d:7770:b132:5823 fe80:0:0:0:0:a447:e853:7ea5:834 Done > masterkey 97e05cb1f2df6f0d9af3026448f4b834 Done > </pre>	<pre>dataset channel 15 Done > > dataset masterkey 97e05cb1f2df6f0d9af3026448f4b834 Done > > dataset commit active Done > > ifconfig up Done > > thread start Done > > state router Done > > </pre>	<pre>state child Done > > state child Done > > panid Duz2136 Done > > ipaddr fd6e:c30c:24e9:e3d9:0:ff:fe00:f001 fd6e:c30c:24e9:e3d9:6fa:ecc5:34b1:27c5 fe80:0:0:0:8065:91ee:3cda:32e Done > ></pre>

router_eligible_2 starts as a child and becomes quickly a router.

Then we can check the capture in the Network Analyzer view.

	2 saved filt	ers /											~ ↓ ③	0	Ē
	0.000s _{11.}		and Assessments and	and with a su	n de dat un	للناه		.1.			ساب		1,816.851s	. 45	0s
	Time:1,816	.8505	59s Real time:N/A	Nodes:10	Event:U[OP N	lessa	ge	PAN filte	er: 213		₹.E	Event Detail	\bigtriangledown	
	^									Frame Control: 0xDC61		^			
L										Frame Type: Data (1)					
L	000440176205 child F001									Security Enabled: false					
L										Frame Pending: false					
L										Ack Required: true					
L	O: 000 state							Intra Pan: true							
L			000 roi)440176317 iter									Frame Version: 2006 (1)		
L			700	0							5		Reserved: 0x00		
┝											· •		Destination Address Mode: Long	(3)	
ŀ	Iransaction	is to	tal:116 shown:116							~	 Levent Detail Event Detail Frame Control: 0xDC61 Frame Type: Data (1) Security Enabled: false Frame Pending: false Ack Required: true Intra Pan: true Frame Version: 2006 (1) Reserved: 0x00 Destination Address Mode: Long (3) Sequence: 0xB9 Destination PAN ID: 0x2136 				
,	Time	Du	Summary	IPv6 Src	IPv6 Dest	P#	М	E#	Error	Warr	^		Sequence: 0xB9		
	1,816	0	MLE Parent Res	fe80::a4	fe80::80	1							Destination PAN ID: 0x2136		~
	1,817	0	MLE Child ID Re	fe80::80	fe80::a4	1						<		>	
1	1 0 1 7	~	the shift is a	6 00 0	6 00 00	~									_

The image above shows communication between the leader (router_eligible_1) and the router (router_eligible_2)

5.6 Remove a leader

We want to discover what happens if we remove the current leader. Unplug the **router_eligible_1** node, wait three minutes and, from each remains nodes enter this command: *state*

ł	Ørouter_eligible_2 ≅	Ø child ≅
	No translation	Solution View Line terminator: CR-LF (DOS, OS/2, MS)
	🚠 Serial 0 🎽 Serial 1 🚔 Admin 🚔 Debug	≧ Serial 0 ≧ Serial 1 ≧ Admin ≧ Debug
	state leader Done	state child Done
	<pre>> child table ID RLOC16 Timeout Age LQ In C_VN R D N Extended MAC 1 0x7c01 240 70 3 96 1 0 0 826591ee3cda032e Done > ipaddr fd6e:c30c:24e9:e3d9:0:ff:fe00:fc00 fd6e:c30c:24e9:e3d9:0:ff:fe00:fc00 fd6e:c30c:24e9:e3d9:be3a:78b9:b377:4de6 fe80:0:0:0:0:808d:e05:4eba:8ca3 Done ></pre>	<pre>> ipaddr fd6e:c30c:24e9:e3d9:0:ff:fe00:7c01 fd6e:c30c:24e9:e3d9:6fa:ecc5:34b1:27c5 fe80:0:0:0:8065:91ee:3cda:32e Done > ></pre>

Once the **router_eligible_1** is removed, the **child** has a new parent, the router (**router_eligible_2**) became the new **leader**, as we can see, in the child table of **router_eligible_2**, there is the extended mac address of the **child**. We can notice some differences for the **child**:

-Only the local address did not change

-Its RLOC16 value changed: 0x7c01

5.7 Put router_eligible_1 back

Plug it back in, open its console and display its state

We need to add it to our network, repeat command lines in the section Add a Node to Our Network.



The router_eligible_1 became a router; the child did not change and router_eligible_2 is still the leader. We notice that Router_eligible_1 does not have anycast address anymore.

5.8 Communication between Nodes

The ping command enables to send a request and check if the communication works, let's have a ping between the leader and the child:

5.8.1 Leader (router_eligible_2) and child ping each other

	Router_eligible_2 console (ping local address)											
Command line	Response	Description										
> ping	ping fe80:0:0:0:8065:91ee:3cda:32e	Sand on ICMDyC Fabo										
	Done	Send an ICIVIPV6 ECHO										
	> 16 bytes from fe80:0:0:0:8065:91ee:3cda:32e: icmp_seq=2	Request										
	hlim=64 time=11ms											
	ping fe80:0:0:0:a447:e853:7ea5:834											

2 saved filters AND								» ⊘ ∉ ∨) 🖻
15.00 p/s					4,810	.406s			1510
	enterteitete entere le transformer								
Time:4,810.405711s Real time:	N/A Nodes:10 Event:ICMPv6 N	Message PAN filter: 2136						💐 Event Detail	
·	•						^	Intra Pan: true	^
000440176317	000440176305	5						Frame Version: 2006 (1)	
7000	(7C05)							Reserved: 0x00	
01	00440176275 sader							Destination Address Mode: Long (3)	
F	000						~	Source Address Mode: Long (3)	
Transactions total:430 shown:	430						⊽ 🗖	Sequence: 0xB2	
Summany	IDv6 Src	IDv6 Dect	D#	N/#	E#	Error Status	Warning A	Destination PAN ID: 0x2136	
		IPVO Dest	P#	IVI#	C#	EITOT Status	warning	Long Destination Address: 828D0E054EBA8CA3	
MLE Child ID Request	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	2					Long Source Address: 826591EE3CDA032E	
MLE Child ID Response	fe80::808d:e05:4eba:8ca3	fe80::8065:91ee:3cda:32e	2					> IEEE 802.15.4 Security [6 bytes]	
Ping	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	4					> 6Lowpan [3 bytes]	
Ping	fe80::a447:e853:7ea5:834	fe80::8065:91ee:3cda:32e	2			Missing pa		ICMP v6 [4 bytes]	
Ping	fe80::808d:e05:4eba:8ca3	fe80::a447:e853:7ea5:834	2					C 3	
MLE Child Update	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	2						
MLE Child Update Response	fe80::808d:e05:4eba:8ca3	fe80::8065:91ee:3cda:32e	2					Hex Dump [58 bytes]	Φ U
MLE Child Update	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	2					8C BA 4E 05 0E 8D 82 2E	^
MLE Child Update Response	fe80::808d:e05:4eba:8ca3	fe80::8065:91ee:3cda:32e	2				\checkmark	03 DA 3C EE 91 65 82 0D	
<							>	0A 00 00 00 01 7A 33 3Az3:	

- 🔅 🛇 🖻

5.8.2 The child and the router (router_eligible_1) cannot ping each other by their local addresses

	child console (ping local address)	
Command line	Response	Description
> ping	ping fe80:0:0:0:a447:e853:7ea5:834 Done	Send an ICMPv6 Echo Request

2 saved filters... AND

-0	.000s	And be also also		and a second strategy of	J J.		4,838	.0578				.779s
Т	ime:4,838.057051s Rea	al time:N/A Nodes:10	Event:ICMPv6 N	lessage PAN filter: 2136					C	3	💐 Event Detail	⊽ □
	•	•							1		Frame Pending: false	^
	000440176317		000440176305								Ack Required: true	
	7000	•****	child (7C05)								Intra Pan: true	
		000440176275 router									Frame Version: 2006 (1)	
		F000								/	Reserved: 0x00	
ī	ransactions total://30	shown:430								3	Destination Address Mode: Long (3)	
	Commence	ID: C Car		ID C Deet	D#	N 4.4	F#	Emer Chatan	Manala a d		Source Address Mode: Long (3)	
	Summary	IPV6 Src		IPv6 Dest	P#	IVI#	E#	Error Status	warning *	1	Sequence: 0x44	
	MLE Child ID Request	fe80::8065:91ee	:3cda:32e	fe80::808d:e05:4eba:8ca3	2						Destination PAN ID: 0x2136	
	MLE Child ID Response	fe80::808d:e05:4	4eba:8ca3	fe80::8065:91ee:3cda:32e	2						Long Destination Address: 826591EE3CDA032E	
	Ping	fe80::8065:91ee	:3cda:32e	fe80::808d:e05:4eba:8ca3	4						Long Source Address: A647E8537EA50834	
	Ping	fe80::a447:e853	:7ea5:834	fe80::8065:91ee:3cda:32e	2			Missing pa			> IEEE 802.15.4 Security [6 bytes]	
	Ping	fe80::808d:e05:4	4eba:8ca3	fe80::a447:e853:7ea5:834	2							, ×
	MLE Child Update	fe80::8065:91ee	:3cda:32e	fe80::808d:e05:4eba:8ca3	2							/

5.8.3 Leader (router_eligible_2) and the router (router_eligible_1) can ping each other by their local addresses

Command line	Response	Description
> ping	<pre>ping fe80:0:0:0:a447:e853:7ea5:834 Done > > 16 bytes from fe80:0:0:0:a447:e853:7ea5:834: icmp_seq=6 hlim=64 time=12m</pre>	Send an ICMPv6 Echo Request

2	saved filters AND									~	≱ ⊘ ⊘	
<mark>17</mark> -0	.00 p/s		Hada A A A A			4,89	8.278s	i			,073.9	07s
T	ime:4,898.277680s Real tim	ne:N/A Nodes:10 Event:ICMPv6	Message PAN filter: 2136							💐 Event Detail	7	7 🗖
	•	٠							^	Frame Version: 2006 (1)		^
	000440176517	00044017630	15							Reserved: 0x00		
	7C00	child (7C05)								Destination Address Mode: Long (3)		
		000440176275								Source Address Mode: Long (3)		
		FOOD							\checkmark	Sequence: 0xC8		
	ransactions total:430 show	m://30						~		Destination PAN ID: 0x2136		
	Summer:	IDu6 See	IDu6 Dest	D#	N/#	F#	Error Status	Marris a	•	Long Destination Address: A647E8537EA5	0834	
	Summary	IPV6 Src	IPV6 Dest	P#	IVI#	E#	Error Status	warning		Long Source Address: 828D0E054EBA8CA	3	
	MLE Child ID Request	fe80::8065:91ee:3cda:32e	te80::808d:e05:4eba:8ca3	2						> IEEE 802.15.4 Security [6 bytes]		
	MLE Child ID Response	fe80::808d:e05:4eba:8ca3	fe80::8065:91ee:3cda:32e	2						> 6Lowpan [3 bytes]		
	Ping	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	4						ICMP v6 [4 bytes]		-
	Ping	fe80::a447:e853:7ea5:834	fe80::8065:91ee:3cda:32e	2			Missing pa			> Application Payload [12 bytes]		
	Ping	fe80::808d:e05:4eba:8ca3	fe80::a447:e853:7ea5:834	2							>	*
	MLE Child Update	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	2								_
	MLE Child Update Response	e fe80::808d:e05:4eba:8ca3	fe80::8065:91ee:3cda:32e	2						Hex Dump [59 bytes]	<	70
	MLE Child Update	fe80::8065:91ee:3cda:32e	fe80::808d:e05:4eba:8ca3	2						F δ 34 69 DC Cδ 36 21 34 .41. 08 A5 7E 53 E8 47 A6 A3~S	.6!4 .G	
	MLE Child Update Response	e fe80::808d:e05:4eba:8ca3	fe80::8065:91ee:3cda:32e	2					~	8C BA 4E 05 0E 8D 82 0DN.		
<								>		FC 03 00 00 01 7A 33 3A 80 00 AA BE 00 07 00 07	.z3:	

The ping request from the child to leader or leader to the router works with the local IPV6 address since there is only one hop!

Let's take a look at the network analyzer trace., There is a "Missing packet" which means a message is missing.

The prefix has a 64 bits length, thread uses IPV6 native address. The MSB 64 bits for the network ID and the LSB 64 bits for the ID device.

We added back **router_eligible_1**, so the topology changed as the RLOC16 of the child too. Once **router_eligible_1** is attached to the network, refresh the console (click on the bin icon) and enter the **rloc16** command to see: 0x7C05.

🎯 ro	outer_eli	gible_1 🛛				
۲	No translation					
2	Serial 0	🚔 Serial 1	🚔 Admin	a Debug		

According to the result of the ping response, we can draw our network topology:



6 Lab2: OpenThread Border Router (Raspberry Pi 3b/b+)

6.1 Prepare your Raspberry Pi

Once you get the image OT_FAE_Training_2020_Labs.img, you have to burn it on an SD card. I recommend using this software to flash your SD card *balenaEtcher-Setup-1.5.100*.

6.2 Prepare 4170A radio board

The OTBR is built with two hardware platforms:

- A host processor hosts the core of OpenThread and the applications, in our case, this is Linux with OpenThread core

- A device controller with the minimal MAC layer: in our case 4170A board with ot-rcp runs on it.

Create an OpenThread project from Simplicity Studio v5 (<u>section 4 to 8 Create an OpenThread Project</u>) and choose *ot-rcp* example, compile it and flash **router_eligible_1**.

6.3 Prepare a node device

Use your child, it has already the corresponding firmware.

6.3.1 OpenThread Border Router setup

At this step you should have this configuration:



4170A board with ot-cli-mtd

Radio board with ot-rcp firmware

6.4 Start the OTBR (OpenThread Border Router)

Once your Raspberry Pi boots,

1. The command line below shows if **otbr-agent** and **otbr-web** processes are running. The expected result is below.

т	92.168.50.128 - pi@raspberrypi: ~ VT
<u>E</u> il	<u>Edit S</u> etup C <u>o</u> ntrol <u>W</u> indow <u>H</u> elp
pi ro ro pi	<pre>saspberryp1: \$ ps aux grep -i otbr 423 0.1 0.3 8856 3764 ? Ss 17:23 0:00 /usr/sbin/othu-agent -I wpan0 spinel+hdlc+uart:///dev/ttyACM ; 424 0.0 0.1 4980 1216 ? Ss 17:23 0:00 /usr/sbin/othu-web 1359 0.0 0.0 4368 576 pts/0 S+ 17:24 0:00 grepcolor=auto -i othr</pre>

- 2. Go to *ot-br-posix* directory: *cd ot-br-posix*
- 3. Run the script sudo./startOTBR_Lab2.sh

💻 192.168.50.128 - pi@raspberrypi: ~/ot-br-posix VT File Edit Setup Control Window Help pi**Praspherrypi:~/ot-br-posix \$** sudo ./startOTBR_Lab2.sh ot-ctl factoryreset sleep 3 ot-ctl thread stop Done sleep 1 ot-ctl ifconfig down Done sleep 1 ot-ctl panid Øxface Done ot-ctl extpanid dead00beef00cafe Done ot-ctl masterkey 00112233445566778899aabbccddeeee Done ot-ctl channel 11 Done ot-ctl networkname SL-OpenThread Done ot-ctl ifconfig up Done ot-ctl thread start Done sleep 5 ot-ctl state leader Done On the client, you will need these commands: dataset channel 11 dataset panid Øxface dataset masterkey ØØ112233445566778899aabbccddeeee dataset commit active ifconfig up thread start

The table below sums up some command lines to setup the OTBR.

	Leader console (startOTBR_Lab2.sh)	
Command-line	Expected Response	Description
> factoryreset		Delete all stored settings, and signal a platform reset
> thread stop	> Done	Disable Thread protocol operation and detach from a Thread network.
> networkname SL-OpenThread	> Done	Set the Thread Network Name.

6.4.1 OTBR info

We can enter commands at the Raspberry Pi terminal:

	Leader console from Raspberry Pi (sudo ot-c	tl)
Command line	Response	Description
> sudo ot-ctl	>	Enter in thread terminal

💻 192.168.50.128 - pi@raspberrypi: ~/ot-br-posix VT

File Edit Setup Control Window Help
piPraspherrypi: */ot-br-posix \$ sudo ot-ctl
> state
leader
Done
> ipaddr
fdde:ad00:beef:0:0:ff:fe00:fc00
fdde:ad00:beef:0:0:ff:fe00:5800
fdde:ad00:beef:0:d39c:92b1:b641:c557
fe80:0:0:0:0:9cd3:2016:4823:f4af
Done

To leave the Thread host terminal hit Ctrl+c keys. Now have a look at our network interface.

The ifconfig Linux command line displays the network interface available.

I92.168.50.128 - pi@raspberrypi: ~ VT
File Edit Setup Control Window Help
<pre>piPraspherrypi: " \$ ifconfig -a eth0: flags=4163<up, broadcast,="" multicast="" running,=""> mtu 1500 inet 192.168.50.128 netmask 255.255.255.0 broadcast 192.168.50.255 inet6 fe80::b6e?:5f9b:5648::d55 prefixlen 64 scopeid 0x20<link/> ether b8:27:eb:25:d5:05 txqueuelen 1000 (Ethernet) RX packets 247 bytes 53618 <52.3 KiB) RX errors 0 dropped 0 overruns 0 frame 0 IX packets 446 bytes 120643 (117.8 KiB) IX errors 0 dropped 0 overruns 0 carrier 0 collisions 0</up,></pre>
<pre>lo: flags=73<up,loopback,running> mtu 65536 inet 127.0.0.1 netmask 255.0.0.0 inet6 ::1 prefixlen 128 scopeid 0x10</up,loopback,running></pre> host> loop txqueuelen 1000 <local loopback=""> RX packets 18 bytes 812 (812.0 B) RX errors 0 dropped 0 overruns 0 frame 0 IX packets 18 bytes 812 (812.0 B) IX errors 0 dropped 0 overruns 0 carrier 0 collisions 0</local>
nat64: flags=4305 <up,pointopoint,running,noarp,multicast> mtu 1500 inet6 fe80::d249:ece6:451:ec59 prefixlen 64 scopeid 0x20<link/> unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-</up,pointopoint,running,noarp,multicast>
<pre>vlan0: flags=4163<up, broadcast,="" multicast="" running,=""> mtu 1500 inet 192.168.50.37 netmask 255.255.255.0 broadcast 192.168.50.255 inet6 fe80::c0e7:8405:2697:6926 prefixlen 64 scopeid 0x20<link/> ether 8e:cd:3c:e5:de:47 txqueuelen 1000 (Ethernet) RX packets 289 bytes 90646 (88.5 KiB) RX errors 0 dropped 0 overruns 0 frame 0 IX packets 183 bytes 33405 (32.6 KiB) IX errors 0 dropped 0 overruns 0 carrier 0 collisions 0</up,></pre>
<pre>wpan0: flags=4305<up,pointopoint,running,noarp,multicast> mtu 1280 inet6 fe80::9cd3:2016:4823:f4af prefixlen 64 scopeid 0x20<link/> inet6 fdde:ad00:beef:0d39c:92b1:b641:c557 prefixlen 64 scopeid 0x0<global> inet6 fdde:ad00:beef::ff:fe00:5800 prefixlen 64 scopeid 0x0<global> inet6 fdde:ad00:beef::ff:fe00:fc00 prefixlen 64 scopeid 0x0<global> unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-</global></global></global></up,pointopoint,running,noarp,multicast></pre>

wpan0 interface was added once the script startOTBR_Lab2.sh was executed successfully

6.4.2 Add a node to our network (out-of-band method)

With the out-of-band method, we know all security information and add the node manually. In the "real world", we use a user interface via the web or mobile app to add the node automatically.

At the end of the script, you can read all the necessary data to add an end device.

```
On the client, you will need these commands:
dataset channel 11
dataset panid Øxface
dataset masterkey Ø0112233445566778899aabbccddeeee
dataset commit active
ifconfig up
thread start
```

From the node, console Add a Node to Our Network the node should join the network properly.



Go back to Raspberry Pi Thread host terminal and check if the child is present in the child table:



So, the child was added successfully, let's ping each other.

6.4.3 Communication between OTBR and the node

The ping command is from the leader $=== \rightarrow$ node

📕 192.168.50.128 - pi@raspberrypi: ~ VT									
<u>F</u> ile <u>E</u> dit <u>S</u> etu	p C <u>o</u> ntrol	Window H	lelp						
pieraspherry	pi:~ \$ suc	do ot-ct	1						
ID RLOC	16 Timeo	out l	Age	LQ	In	C_UN	IRIDINI	Extended MA	IC I
¦ 1 ¦ 0×58	01	240	201		3	: 35	1110101	c627f713987	28122
Done > ipaddr fdde:ad00:be fdde:ad00:be fdde:ad00:be fe80:0:0:0:9 Done > ping fe80: Done > 16 bytes f	eef:0:0:ff eef:0:0:ff ef:0:d39c cd3:2016:4 0:0:0:c42 crom fe80:0	:fe00:fc :fe00:58 :92b1:b6 4823:f4a; 7:f713:9; 0:0:0:c4;	00 90 41:c557 f 872:8122 27:f713:9872	:812	2: i	cmp_sec	q=1 hlim	=64 time=34m	าร

Go back to your child console, ping from the node === \rightarrow leader

> ipaddr fdde:ad00:beef:0:0:ff:fe00:5801 fdde:ad00:beef:0:452e:1161:341a:fca4 fe80:0:0:0:c427:f713:9872:8122 Done > > ping fe80:0:0:0:9cd3:2016:4823:f4af Done > > 16 bytes from fe80:0:0:0:9cd3:2016:4823:f4af: icmp_seq=1 hlim=64 time=32ms

Communication through the local address works on both sides.

7 Demo

In this demonstration, the user will discover the commissioning with a user interface command line. The commissioner application is embedded in the Raspberry Pi contrary to the external commissioner, which is a web or mobile application. All steps will be described, I will show you how to automatize all steps.

OpenThread Commissioner

A Thread Commissioner connects to and manages a Thread network. A Thread network requires a Commissioner to commission new Joiner devices.



We need an OTBR and another node, the software/hardware configuration is the same as the Lab2.

Restart your Raspberry Pi, make sure the node has ot-cli-mtd firmware.

Go to ot-commissioner folder: cd ot-commissioner

Execute the following script: sudo ./scriptCommissionDemo.sh

All-new command lines are described below (all characters between "\${}" are script bash variable):

```
iPraspherrypi: "/ot-commissioner $ sudo ./scriptCommissionDemo.sh
ot-ctl factoryreset
sleep 3
ot-ctl thread stop
Done
sleep 1
ot-ctl ifconfig down
Done
sleep 1
ot-ctl panid Øxface
Done
ot-ctl extpanid dead00beef00cafe
Done
ot-ctl masterkey 00112233445566778899aabbccddeeee
Done
ot-ctl channel 11
Done
ot-ctl networkname SL-OpenThread
Done
<del>××××</del>SL-OpenThread
<del>××××</del>dead00beef00cafe
____SL-OpenThread J01NME dead00beef00cafe
Done
****pskc is dbc1dc72993de43bfb2bd5f853d3ec63
ot-ctl ifconfig up
Done
ot-ctl prefix add 2001:dce:1:ffff::/64 pasor
Done
ot-ctl thread start
Done
ot-ctl state
leader
Done
dbc1dc72993de43bfb2bd5f853d3ec63
Done
/usr/local/etc/commissioner/non-ccm-config.json dbc1dc72993de43bfb2bd5f853d3ec63
File name modified is /usr/local/etc/commissioner/non-ccm-config.json
New PSKc is dbc1dc72993de43bfb2bd5f853d3ec63
```

Leader console from	Raspberry Pi (sudo ot-ctl)	
Command-line	Expected Response	Description
> ot-ctl extpanid dead00beef00cafe	> Done	Set the Thread Extended PAN ID value.
> ot-ctl pskc -p J01NME \${EXTPANID} \${NETWORKNAME}	> Done	Generate a hex-encoded PSKc by using a Passphrase (Commis- sioner Credential), the extpanid, and the Network Name with the PSKc Generator tool on the OTBR. Make sure to use the same Extended PAN ID and Network Name that was used in the operational dataset:
ot-ctl prefix add 2001:dce:1:ffff::/64 pasor		Add a valid prefix to the Network Data.

		In other words, the OTBR and the node bust have the same network ID, to be on the same network.
ot-ctl pskc	> Done dbc1dc72993de43bfb2bd5f85 3d3ec63	Display security key
./updatePSKc.sh \${PATH_PSKC} \${PSKC}		Replace PSKC in /usr/lo- cal/etc/commissioner/non- ccm-config.json and change the PSKc

We can see our new pskc in /usr/local/etc/commissioner/non-ccm-config.json the command line below can show quicker the result:

cat /usr/local/etc/commissioner/non-ccm-config.json | grep -i pskc

iPraspberrypi:"/ot-commissioner \$ cat /usr/local/etc/commissioner/non-ccm-config.json | grep -i pskc "PSKc" : "dbc1dc72993de43bfb2bd5f853d3ec63"

Now we can start our commissioner application:

Open a new terminal from your Raspberry Pi and enter the followings commands below:

- 1. commissioner-cli /usr/local/etc/commissioner/non-ccm-config.json
- 2. start :: 49191
- 3. active
- 4. joiner enableall meshcop J01NU5

Leader console from	Raspberry Pi (sudo ot-ctl)	
Command line	Response	Description
commissioner-cli /usr/local/etc/commis- sioner/non-ccm-config.json	A commissioner shell opens	Start the OT Commissioner CLI with the Non-CCM configuration:
> start :: 49191	[done]	Connect to OTBR
> active	true [done]	Verify that the Commissioner is active
> joiner enableall meshcop J01NU5	> Done	In OT Commissioner, enable Thread 1.1 MeshCoP joiner for all Joiners with a password of J01NU5:



7.1.1 Join the network

From the joiner (node device), enter the following commands lines (image below), it should take 2 minutes:

As we can see, at the beginning the device is not in the network.

♣ ot-rcp.slcp 🔗 node_child (440176275) 🛛							×									
No translation									2	~	Line	termi	nator:	CR-LF	(DOS, C	
2	Serial 0	🛋 S	erial '	2	Admin	<u></u> [Debu	g								
<pre>sta dis Dor > i Dor > j Dor > t Dor > t Dor > s chi Dor</pre>	ate sabled ne joiner s ne thread s ne state ild ne	f up start start	: J01	1115												
> = 1 4 2	10 01 03 41 44 22 2E 32 20 00 00 10	==[[1 2 21 2 05 0 47)	HCI] OD 5 45 4 69 7	dir 3 4C 5 52 4 48 	ection 2D 33 75 	=send 4F 50 32 23 62 2D	t 45 10 66	ype 4E 31 25 (=J0 54 2E 06	IN_ 48 30 18	FIN 52 2E B4	1.req 45 30 30 	le ! AD". .2 G	n=051 .SL-OF EFR32# itHub	===== PENTHRE #.1.0.0 -f%40	
 1 [Tŀ Joi	IC 01 0: HCI] din HCI] din in succe	(THO recti recti	on=r	irec ecv end	tion=ro type type	∍cv -JOIN -JOIN	typ _ENT _ENT	e=J(.nti .rsp	OIN f p	FI	N.1	rsp 	len=	003]		
0																

We do not need to enter channel, masterkey, panid and so on, all steps were automatized in bash script.

In case you have a fail message, repeat joiner start J01NU5 command line.

8 Communicate with the external world

Previously we added a prefix to communicate locally with IPV6 addresses:

	OTBR pings child (ping local address)	
Command line	Response	Description
> ping	<pre>sudo ot-ctl > ping fe80:0:0:0:584b:fe7e:55d3:61fd Done > 16 bytes from fe80:0:0:0:584b:fe7e:55d3:61fd: icmp_seq=1 hlim=64 time=37ms</pre>	Send an ICMPv6 Echo Reques
	child pings OTBR (ping local address)	
Command line	Response	Description
> ping	<pre>> ping fe80::e01c:c24:102:dc9 Done > > 16 bytes from fe80:0:0:0:e01c:c24:102:dc9: icmp_seq=4 hlim=64 time=32ms</pre>	Send an ICMPv6 Echo Reques

In both directions, the local address works, since there is only one hop.

If I want to communicate with my IPV6 *eth0* interface, it will not work, the mesh prefix was not created for it. The interface *eth0* uses ipv4 address, to communicate between IPV6 and IPV4, we need a translater: NAT64.

eth0: flags=4163<UP,BROADCAST,RUNNING, MILTĪCAST mtu 1500 255.255.0 broadcast 192.168.50.255 prefixlen 64 scopeid 0x20<link> inet 192.168.50.128 inet6 fe80::b6e7:5f 255.255.255.0 netmask 5648:ed55 inet6 1000 b8:27:eb:25:d5:05 txaueuelen (Ethernet) ether bytes packets 1296 <126.5 KiB> 1075 1 И И dropped И errors overruns rame 90 2488(243 0 KiB) packets bytes errors Ø 5 carrier Ø collisions Ø dropped overruns Ø

Contrary to eth0, wpan0 interface has the mesh prefix

pieraspherrypi:"/ot-commissioner \$ ifconfig wpan0
wpan0: flags=4305 <up,pointopoint,running,nöarp,multicast> mtu 1280</up,pointopoint,running,nöarp,multicast>
inet6 fdde:ad00:beef::ff:fe00:4c00 prefixlen 64 scopeid 0x0 <global></global>
inet6 fdde:ad00:beef:0:5fc9:3d24:caa8:f611 prefixlen 64 scopeid 0x0 <global></global>
inet6 fe80::e01c:c24:102:dc9 prefixlen 64 scopeid 0x20 <link/>
inet6_fdde_ad00_beef;ff:fe00:fc00_prefixlen_64_scopeid_0x0 <global></global>
inet6_2001:dce:1:ffff:60de:a6a9:1019:7812_prefixlen_64_scopeid_0x0 <global></global>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-
RX packets 8 bytes 448 (448.0 B)
RX errors Ø dropped 1 overruns Ø frame Ø
TX packets 21 bytes 2936 (2.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

In /etc/tayga.conf, we have the prefix to communicate between IPV4 and IPV6,.

The prefix used in NAT64 is 2001:db8:1:ffff::/96, you can retrieve it with the following command line:

cat /etc/tayga.conf | grep -i prefix.*96\$

Do not mix NAT64 prefix with the mesh prefix, the mesh prefix 2001:dce:1:ffff is used to communicate in mesh network and the prefix NAT64 2001:db8:1:ffff::/96 is used to communicate between IPV4 and IPV6 addresses.

💻 192.168.50.128 - pi@raspberrypi: ~/ot-commissioner VT													
File	Edit	Setup	Control	Window	Help								
piP	aspb	errypi	:~⁄ot-	co nn issi <mark>Hee/36</mark>	ioner	\$ cat	/etc/tag	yga.conf	ł	grep	-i	prefix.*	96\$
prefix 64:ff9b==796 pieraspberrypi:~/ot-commissioner \$													

To ping an IPV4, just use the NAT64 prefix and add the IPV4 address in hex format, here we have 192.168.50.128

Decimal	Hexadecimal	Prefix::IPV4	Description
192.168.50.128	C0a8:3280	2001:db8:1:ffff <mark>::</mark> C0a8:3280	Communicate with eth0 interface

Child pings (eth0 interface)								
command	Response	Description						
ping 2001:db8:1:ffff::C0a8:3280	<pre>> 16 bytes from 2001:db8:1:ffff:0:0:c0a8:3280: icmp_seq=8 hlim=62 time=42ms Done</pre>	ping eth0 address						
Child pings google								
ping 2001:db8:1:ffff::808:808 Done	<pre>> 16 bytes from 2001:db8:1:ffff:0:0:808:808: icmp_seq=9 hlim=116 time=50ms</pre>	ping google						

Here is the result of the external communication of the child and the external world, you can get IPV4 of Google from this link <u>https://fr.wik-ipedia.org/wiki/Google_Public_DNS</u>

Google Public DNS

Google Public DNS is a service from Google that provides recursive DNS servers to Internet users. It was announced onDecember 9, 2009¹.

The anycast IP addresses of the servers are as follows:

- IPv4: 8.8.8.8 and 8.8.4.4
- IPv6: 2001: 4860: 4860 :: 8888 and 2001: 4860: 4860 :: 8844

 0	t-rcp.slcp	🦃 nod	le_child (44	0176275) 🛛					
9	No translation					Line	terminator:	CR-LF	(DOS, OS
🛎 Serial 0 🚔 Serial 1 🚔 Admin 🚔 Debug									
<pre>ipaddr ipaddr fdde:ad00:beef:0:0:ff:fe00:4c02 fe80:0:0:0:584b:fe7e:55d3:61fd 2001:dce:1:ffff:b407:8914:b223:3513 fdde:ad00:beef:0:452e:1161:341a:fca4 Done > > > > > > > > > > > > > > > > > > ></pre>									
>	16 bytes fr	om 200	1:db8:1:ff	ff:0:0:808:	808: ic	mp_seq:	=6 hlim=116	5 time=	47ms
\	Σ								