Nouvelle version de l'outil EvalVSN : OpenCV 2.4.9 + Ubuntu 14.04

I. Objectif :

Porter le code en C++ en utilisant OpenCV et étudier la faisabilité et juger la difficulté de développement en utilisant OpenCV, pour permettre une meilleure portabilité par rapport au code MATLAB ainsi que la possibilité d'interagir avec des simulateurs basés sur C++ comme NS2 [1] et Castalia [2], et faire des tests sur la plateforme IoT-LAB [3] en utilisant Contiki

II. Structure du dossier de l'outil EvalVSN :

Afin de bien organiser les sources d'EvalVSN, on propose cette structure :

1. À la racine : un fichier CMakeLists.txt qui construit un Makefile approprié pour la compilation. Cette méthode permet une compilation/installation avec un simple cmake/make pour l'utilisateur au lieu de disposer d'un IDE particulier

2. Un répertoire src : contenant les fichiers sources *.cc à compiler, l'exécutable sera généré dans le répertoire bin

3. Un répertoire bin : contient l'exécutable evalvsn après compilation des fichiers sources *.cc

4. Un répertoire scripts : contient les différents fichiers scripts qui facilitent l'usage de l'outil evalvsn

5. Un répertoire ipFiles : contient des modèles de [ipFile] pour un premier test, correspondant au premier argument *input_parameters_file* de l'exécutable evalvsn

6. Un répertoire vidéo : contenant des exemples de vidéos *.yuv [4] [5] pour un premier test, qui doit être compatible avec l'outil avconv pour la conversion de cette video vers une vidéo au format avi

III. Arguments de l'exécutable evalvsn :

Usage : evalvsn [input_parameters_file] [video_file] [output_directory]

Arguments :

1. *input_parameters_file* : chemin absolu ou relative du ficher qui contient les paramètres d'entrée

2. *video_file* : chemin absolu ou relative du fichier video au format avi

3. *output_directory* : c'est le répertoire crée qui contiendra tous les fichiers et sous-répertoire crées par l'outil evalvsn, son nom suit le même principe qu'un nom de dossier valide sous linux.

IV. Format et paramètres du ficher *input_parameters_file* :

1. Format codage et décodage (Référence) sans transmission réseau :

#|simId|Encoder|imageResolution|imagesNb|videoDuration|FPS|Threshold|gopCoef|qualityCoef|priorityNb|DCT|zoneSize|rleLevel|binLevel|rcvFile|
CLA8 MMPEG 144x176 300 12 1 0 1 1.1 1 CLA 8 RLE2 EG

2. Format décodage avec transmission réseau :

#|simId|Encoder|imageResolution|imagesNb|videoDuration|FPS|Threshold|gopCoef|qualityCoef|priorityNb|DCT|zoneSize|rleLevel|binLevel|rcvFile|
CLA8 MMPEG 144x176 300 12 1 0 1 1.1 1 CLA 8 RLE2 EG ./evalvsnDir/hall_qcif-CLA8/rt-packet.dat

3. Paramètre :

Paramètre	Valeurs possibles	Fonctionnalité
simId	Un nom de dossier valide sous linux, ex : 1, 2, CLA8, etc	Sert à déterminer le nom du dossier de la simulation comme suit : *video_file* = ../video/hall_qcif.avi + *simId* = CLA8 ⇒ Nom du dossier de la simulation = hall_qcif-CLA8
Encoder	MMPEG, SC	Sert au codage et décodage de la vidéo [video_file] dans l'outil evalvsn avec : MMPEG : codage avec compression SC : codage sans compression
imageResolution	32x32, 128x128, 144x176, 512x512, etc	Sert à redimensionner l'image de la vidéo [video_file] comme suit : *imageResolution* = 144x176 ⇒ hauteur = 144 et largeur = 176
imagesNb	300, 2000, etc	Nombre total d'images de la video [video_file]
videoDuration	10, 12, 60, etc	Durée total en secondes de la video [video_file]
FPS	1, 2, 15, 30, etc	FPS de la video [decodedVideo.avi] après décodage
Threshold	$[0,+\infty[$	Sert à diminuer le nombre de frames à coder dans l'étape de codage et copier la dernière frame reçu correctement dans l'étape de décodage
gopCoef	$[0,1]$	Sert à déterminer si une frame doit être codée en M-Frame ou D-Frame
qualityCoef	$]0,+\infty[$	Sert à ajuster la qualité souhaitée de la vidéo
priorityNb	$[1,13]$	Sert à déterminer le nombre de niveaux de priorité par block
DCT	CLA, LLM, BIN	Sert à appliquer une DCT classique (CLA) ou une DCT triangulaire (LLM ou BIN)
zoneSize	$[2,8]$	Sert à réduire la complexité de l'étape DCT, en termes de nombre d'opérations nécessaire (addition, multiplication, décalage) avec une DCT zonale rapide

rleLevel	RLE0, RLE1, RLE13, RLE30, etc	Sert à déterminer si un niveau de priorité doit être codé sans ou avec RLE comme suit :
		rleLevel = RLE2 + *priorityNb* = 1 ⇒ *rleLevel* = 2 et priorité de tous les niveaux = 1 ⇒ coder tous les niveaux de priorité 1 sans RLE, suivant cette algorithme :
		IF (priority(Level) >= rleLevel) THEN
		Encoding Level with RLE
		ELSE
		Encoding Level without RLE
		ENDIF

binLevel	ST, EG	Sert au codage binaire du code sans ou avec RLE : ST : codage binaire statique EG : codage binaire Exponential-Golomb
rcvFile	./evalvsnDir/hall_qcif-CLA8/rt-packet.dat, etc	Sert à déterminer le chemin absolu ou relative du fichier de trace réception lorsqu'on souhaite faire du décodage avec transmission réseau

4. Si le paramètre *Encoder* = SC alors :

   a. Format codage et décodage (Référence) sans transmission réseau :

   #|simId|Encoder|imageResolution|imagesNb|videoDuration|FPS|Threshold|priorityNb|rcvFile|
   1 SC 144x176 300 10 6 0 1

   b. Format décodage avec transmission réseau :

   #|simId|Encoder|imageResolution|imagesNb|videoDuration|FPS|Threshold|priorityNb|rcvFile|
   1 SC 144x176 300 10 6 0 1 ../simNS/set6/hall_qcif-1/sim1/rt-packet.dat

V. Fichiers scripts (rép. scripts)

Fichier script	Fonctionnalité	Arguments
generateStPacket.sh	Script shell qui génère le fichier de trace par paquet st-packet.dat	Usage : bash generateStPacket.sh [stPath/st-priority.dat] [maxPayload] 1. *stPath/st-priority.dat* : chemin absolu ou relative du fichier st-priority.dat 2. *maxPayload* : taille maximale de la charge du paquet en octets
ns.sh	Script shell qui permet de lancer une transmission réseau sur le simulateur NS2	Usage : bash ns.sh [st-packet.dat] [rtName] [videoDuration] [nn] [ifq] 1. *st-packet.dat* : fichier de trace par paquet 2. *rtName* : nom du fichier de trace réception à générer par le script tcl, comme suit : *rcvFile* = ./evalvsnDir/hall_qcif-CLA8/rt-packet.dat ⇒ *rtName* = rt-packet.dat 3. *videoDuration* : durée total en secondes de la video [video_file] 4. *nn* : nombre de nœud de la simulation, ex. 50, 100, 225, 400, etc 5. *ifq* : interface de la file d'attente du nœud dans la simulation, ex. Queue/drop-tail, Queue/mmpeg_priqueue, Queue/sc_priqueue, etc
evalvsn.tcl	Script tcl qui permet de simuler une transmission réseau sur le simulateur NS2	Usage : ns evalvsn.tcl -st [st-packet.dat] -rt [rtName] -simt [videoDuration] -nn [nn] -ifq [ifq]
iotlab.sh	Script shell qui permet de lancer une transmission réseau sur la plateforme IoT-LAB en utilisant Contiki	Usage : bash iotlab.sh [experimentName] [siteName] [userName] [password] [sourceNode] [sinkNode] [resourceidList] [firmwareName] 1. *experimentName* : nom de l'expérience, ex. evalvsn, etc 2. *siteName* : nom du site, ex. grenoble, etc 3. *userName* : nom de l'utilisateur du compte IoT-LAB, ex. baziz, etc 4. *password* : mot de passe de l'utilisateur du compte IoT-LAB, ex. evalvsn, etc 5. *sourceNode* : nœud source, ex. 26, etc 6. *sinkNode* : nœud de destination ou sink, ex. 39, etc 7. *resourceidList* : liste des nœuds, source, voisin et destination, ex. 26+30+34+39, etc 8. *firmwareName* : nom du firmware binaire, ex. evalvsn.ihex, etc
evalvsn.ihex	Firmware binaire qui permet de réaliser une transmission réseau multi-saut sur la plateforme IoT-LAB en utilisant Contiki	N/A
set.sh	Script shell qui permet de lancer une transmission réseau soit sur le simulateur NS2 ou sur la plateforme IoT-LAB en utilisant Contiki	Usage NS2 : bash set.sh [input_parameters_file] [video_file] [output_directory] [maxPayload] [nn] [ifq] ou Usage IoT-LAB : bash set.sh [input_parameters_file] [video_file] [output_directory] [maxPayload] [experimentName] [siteName] [userName] [password] [firmwarePath] [sourceNode] [sinkNode] ou Usage IoT-LAB : bash set.sh [input_parameters_file] [video_file] [output_directory] [maxPayload] [experimentName] [siteName] [userName] [password] [firmwarePath] [sourceNode] [neighborNode] [sinkNode] 9. *firmwarePath* : chemin absolu ou relative du ficher firmware binaire, ex. evalvsn.ihex, etc 11. *neighborNode* : nœuds voisins, ex. 30+34, etc
sets.sh	Script shell qui permet de lancer plusieurs sets séquentiellement	Usage : bash sets.sh 1. Conversion de la vidéo hall_qcif.yuv de résolution qcif vers [video_file] avconv -y -s qcif -i hall_qcif.yuv [video_file] 2. Codage et décodage de référence sans transmission réseau, avec *input_parameters_file* = encoder.dat evalvsn [input_parameters_file] [video_file] [output_directory] 3. Transmission réseau sur le simulateur NS2, avec *input_parameters_file* = decoderNS.dat bash set.sh [input_parameters_file] [video_file] [output_directory] [maxPayload] [nn] [ifq] 4. Transmission réseau sur la plateforme IoT-LAB en utilisant Contiki, avec *input_parameters_file* = decoderIOTLAB.dat bash set.sh [input_parameters_file] [video_file] [output_directory] [maxPayload] [experimentName] [siteName] [userName] [password] [firmwarePath] [sourceNode] [neighborNode] [sinkNode] 5. Décodage de la transmission réseau sur le simulateur NS2, avec *input_parameters_file* = decoderNS.dat evalvsn [input_parameters_file] [video_file] [output_directory] 6. Décodage de la transmission réseau sur la plateforme IoT-LAB en utilisant Contiki, avec *input_parameters_file* = decoderIOTLAB.dat evalvsn [input_parameters_file] [video_file] [output_directory]
nsStats.sh	Script shell qui permet de réaliser des statistiques sur la simulation NS2	Usage : bash nsStats.sh st-packet.tr 1. *st-packet.tr* : fichier de trace NS2
nsEvalVSN.patch	Patch de mise à jour du simulateur NS2	Usage : cd ~/ns-allinone-2.35/ns-2.35 & patch -p2 < nsEvalVSN.patch
iotlabEvalVSN.patch	Patch de mise à jour de la plateforme IoT-LAB en utilisant Contiki	Usage : cd ~/iot-lab/parts/wsn430/OS/Contiki/examples & patch -p10 < ~/evalvsn/scripts/iotlabEvalVSN.patch

VI. Répertoire, sous-répertoires et fichiers créés par l'outil evalvsn :

Processus	Entrées	Sorties			
		Répertoire	Sous-répertoires		Fichiers
Conversion de la vidéo hall_qcif.yuv vers [video_file], avec :	[hall_qcif.yuv]	N/A	N/A		[video_file]
Codage et décodage de référence sans transmission réseau, avec : *video_file* = hall_qcif.avi *simId* = 1	[input_parameters_file] [video_file] [output_directory]	[output_directory]	Niveau 1	[hall_qcif-1]	[st-frame.dat] [st-priority.dat]
			Niveau 2	[allImages]	hall_qcif_*.png
				[capturedImages]	[capturedVideo.avi] frame_*.png
				[referenceImages]	[referenceVideo.avi] frame_*.png
Transmission réseau sur le simulateur NS2, avec : *video_file* = hall_qcif.avi *simId* = 1 *rcvFile* = ../simNS/set1/hall_qcif-1/ns2/rt-packet.dat	[input_parameters_file] [video_file] [output_directory] [maxPayload] [nn] [ifq]	[simNS]	Niveau 1	[set1]	N/A
			Niveau 2	[hall_qcif-1]	N/A
			Niveau 3	[ns2]	[st-packet.tr] [st-packet.nam] [ns2_st-packet.dat] [rt-packet.dat]
Transmission réseau sur la plateforme IoT-LAB en utilisant Contiki, avec : *video_file* = hall_qcif.avi *simId* = 1 *rcvFile* = ../simIOTLAB/set1/hall_qcif-1/iotlab/rt-packet.dat	[input_parameters_file] [video_file] [output_directory] [maxPayload] [experimentName] [siteName] [userName] [password] [firmwarePath] [sourceNode] [neighborNode] [sinkNode]	[simIOTLAB]	Niveau 1	[set1]	N/A
			Niveau 2	[hall_qcif-1]	N/A
			Niveau 3	[iotlab]	[site.dat] [nodes.dat] [experiment.dat] [rt-packet.dat]
Décodage de la transmission réseau sur le simulateur NS2, avec : *video_file* = hall_qcif.avi *simId* = 1 *rcvFile* = ../simNS/set1/hall_qcif-1/ns2/rt-packet.dat	[input_parameters_file] [video_file] [output_directory]	[output_directory]	Niveau 1	[hall_qcif-1]	[st-packet.dat]
			Niveau 2	[ns2]	[mt.dat] [rt-frame.dat] [rt-packet.dat]
			Niveau 3	[decodedImages]	[decodedVideo.avi] [frame_*.png]
Décodage de la transmission réseau sur la plateforme IoT-LAB en utilisant Contiki, avec : *video_file* = hall_qcif.avi *simId* = 1 *rcvFile* = ../simNS/set1/hall_qcif-1/iotlab/rt-packet.dat	[input_parameters_file] [video_file] [output_directory]	[output_directory]	Niveau 1	[hall_qcif-1]	[st-packet.dat]
			Niveau 2	[iotlab]	[mt.dat] [rt-frame.dat] [rt-packet.dat]
			Niveau 3	[decodedImages]	[decodedVideo.avi] [frame_*.png]

VII. Format des fichiers générer par l'outil evalvsn :

1. Format du fichier [st-frame.dat] :

frameNb	frameType	frameSize	refPSNR	refSSIM	compRatio
1	M	7221	35.8442	0.963857	350.976
2	D	3283	34.92	0.955687	771.977
3	D	3231	34.8689	0.954167	784.401
…					

- frameNb : numéro de la frame codée
- frameType : type de la frame codée
- frameSize : taille en octets de la frame codée
- refPSNR : PSNR de référence en décibels de la frame décodée
- refSSIM : SSIM de référence de la frame décodée
- compRatio : ratio de compression en pourcentages de la frame décodée

2. Format du fichier [st-priority.dat] :

sendTime	frameNb	frameType	dataPriority	dataSize	data
0	1	M	1	7221	2FE7EFCABEABF02E6FA2F82DBBBF1AFC0B9CDBCA …
1	2	D	1	3283	376CB72B6CB72B3AACFCDADAB6CE6B7F56B2DFDE …
2	3	D	1	3231	B2F2CADBCAEADCCB6ACAB3F46CDEEB3FB6AE6EAC …
…					

- sendTime : temps de transmission en seconds des niveaux de priorité codés avec le même type et priorité
- frameNb : numéro de la frame codée des niveaux de priorité codés avec le même type et priorité
- frameType : type de la frame codée des niveaux de priorité codés avec le même type et priorité
- dataPriority : priorité des niveaux de priorité codés avec le même type et priorité
- dataSize : taille en octets des niveaux de priorité codés avec le même type et priorité
- data : données en hexadécimales des niveaux de priorité codés avec le même type et priorité

3. Format du fichier [st-packet.dat] :

sTime	seqNb	pSize	frameNb	frameType	prioNb	offset
0	0	50	1	M	1	0
0	1	50	1	M	1	50
0	2	50	1	M	1	100
…						

- sTime : temps de transmission en seconds du paquet
- seqNb : numéro de séquence du paquet
- pSize : taille en octets du paquet
- frameNb : numéro de la frame codée du paquet
- frameType : type de la frame codée du paquet
- prioNb : priorité du paquet
- offset : offset du paquet

4. Format du fichier [mt.dat] :

psnr	refPSNR	ssim	refSSIM	pktLoss	pktLossM	pktLossD	pktlossM1	pktlossM2	pktlossM3	…	pktlossM13	pktlossD1	pktlossD2	pktlossD3	…	pktlossD13
31.3566	34.8705	0.939425	0.956173	27.1077	0	33.3866	0	na	na	…	na	33.3866	na	na	…	na

- psnr : moyenne des PSNR des frames décodées
- refPSNR : moyenne des PSNR de référence des frames décodées
- ssim : moyenne des SSIM des frames décodées
- refSSIM : moyenne des SSIM de référence des frames décodées
- pktLoss : moyenne des taux de pertes de paquets global des frames décodées
- pktLossM : moyenne des taux de pertes de paquets de type M des frames décodées
- pktLossD : moyenne des taux de pertes de paquets de type D des frames décodées
- pktlossM1 : moyenne des taux de pertes de paquets de type M et de priorité 1 des frames décodées
- pktlossM2 : moyenne des taux de pertes de paquets de type M et de priorité 2 des frames décodées
- pktlossM3 : moyenne des taux de pertes de paquets de type M et de priorité 3 des frames décodées

- pktlossM13 : moyenne des taux de pertes de paquets de type M et de priorité 13 des frames décodées
- pktlossD1 : moyenne des taux de pertes de paquets de type D et de priorité 1 des frames décodées
- pktlossD2 : moyenne des taux de pertes de paquets de type D et de priorité 2 des frames décodées
- pktlossD3 : moyenne des taux de pertes de paquets de type D et de priorité 3 des frames décodées
- pktlossD13 : moyenne des taux de pertes de paquets de type D et de priorité 13 des frames décodées

5. Format du fichier [rt-frame.dat] :

frameNb	psnr	ssim	pktLoss	pktloss1	pktloss2	pktloss3	pktloss4	pktloss5	pktloss6	pktloss7	pktloss8	pktloss9	pktloss10	pktloss11	pktloss12	pktloss13
1	35.8442	0.963857	0	0	na	na	na	na	na	na	na	na	na	na	na	na
2	23.0631	0.901279	86.3636	86.3636	na	na	na	na	na	na	na	na	na	na	na	na
3	23.1031	0.903208	100	100	na	na	na	na	na	na	na	na	na	na	na	na
…																

- frameNb : numéro de la frame décodée
- psnr : PSNR de la frame décodée
- refPSNR : PSNR de référence de la frame décodée
- ssim : SSIM de la frame décodée
- refSSIM : SSIM de référence de la frame décodée
- pktLoss : taux de pertes de paquets global de la frame décodée
- pktLossM : taux de pertes de paquets de type M de la frame décodée
- pktLossD : taux de pertes de paquets de type D de la frame décodée
- pktlossM1 : taux de pertes de paquets de type M et de priorité 1 de la frame décodée
- pktlossM2 : taux de pertes de paquets de type M et de priorité 2 de la frame décodée
- pktlossM3 : taux de pertes de paquets de type M et de priorité 3 de la frame décodée
- pktlossM13 : taux de pertes de paquets de type M et de priorité 13 de la frame décodée
- pktlossD1 : taux de pertes de paquets de type D et de priorité 1 de la frame décodée
- pktlossD2 : taux de pertes de paquets de type D et de priorité 2 de la frame décodée

- pktlossD3 : taux de pertes de paquets de type D et de priorité 3 de la frame décodée
- pktlossD13 : taux de pertes de paquets de type D et de priorité 13 de la frame décodée

6. Format du fichier [rt-packet.dat] (créer par NS2 ou IoT-LAB) :

sTime	seqNb	pSize	frameNb	frameType	prioNb
0	0	50	1	M	1
0	1	50	1	M	1
0	2	50	1	M	1
…					

- sTime : temps de réception en seconds du paquet reçu
- seqNb : numéro de séquence du paquet reçu
- pSize : taille en octets du paquet reçu
- frameNb : numéro de la frame codée du paquet reçu
- frameType : type de la frame codée du paquet reçu
- prioNb : priorité du paquet reçu
- offset : offset du paquet reçu

VIII. Installation de l'outil evalvsn :

1. En supposant que vous travaillez dans votre répertoire personnel :

   $ cd ~

2. Télécharger et installer OpenCV (version 2.4.9 ou supérieur) :

   Dépend de la distribution Linux que vous utilisez. Pour Debian (et Ubuntu), vous pouvez utiliser apt-get ou ce lien [6].

3. Télécharger et installer avconv :

   $ sudo apt-get install libav-tools

4. Télécharger evalvsn.tar.gz
5. Décompresser evalvsn.tar.gz :

   $ tar -xvzf evalvsn.tar.gz

$ cd evalvsn

6. Compilation des fichiers sources *.cc présent dans le répertoire src afin de générer l'exécutable evalvsn dans le répertoire bin :

   $ cmake .

   $ make

IX. Installation du simulateur NS2 avec l'outil evalvsn :

1. Télécharger et installer le simulateur NS2 (version ns-allinone-2.35.tar.gz) :

   Pour Debian (et Ubuntu), vous pouvez suivre ce lien [7].

2. Mise à jour du simulateur NS2 avec les nouveaux agents de transmission de paquets :

   $ cd ~/ns-allinone-2.35/ns-2.35/

   $ patch -p2 < ~/evalvsn/scripts/nsEvalVSN.patch

   $ ./configure

   $ make clean

   $ make

X. Installation de la plateforme IoT-LAB avec l'outil evalvsn :

1. Configuration de l'environnement de la plateforme IoT-LAB en utilisant Contiki :

   $ git clone https://github.com/iot-lab/iot-lab.git

   $ cd ~/iot-lab

   $ make

   $ make setup-wsn430

2. Mise à jour de la plateforme IoT-LAB en utilisant Contiki avec les nouveaux agents de transmission de paquets :

   $ cd ~/iot-lab/parts/wsn430/OS/Contiki/examples

   $ patch -p10 < ~/evalvsn/scripts/iotlabEvalVSN.patch

   $ make clean

$ make evalvsn RADIO=WITH_CC1101

XI. Exécution de l’exemple fournie dans l’archive evalvsn.tar.gz :

$ cd ~

$ tar -xvzf evalvsn.tar.gz

$ cd evalvsn/scripts

$ bash sets.sh

REFERENCES

[1] www.isi.edu/nsnam/ns/
[2] https://castalia.forge.nicta.com.au/index.php/en/
[3] https://www.iot-lab.info.
[4] http://www2.tkn.tu-berlin.de/research/evalvid/cif.html
[5] http://www2.tkn.tu-berlin.de/research/evalvid/qcif.html
[6] http://www.samontab.com/web/2014/06/installing-opencv-2-4-9-in-ubuntu-14-04-lts/
[7] https://www.howtoforge.com/tutorial/ns2-network-simulator-on-ubuntu-14.04/