



# Implémentation FPGA d'algorithmes de surveillance de trafic

*Projet 11 :*

Benoît FONTAINE

Tristan GROLÉAT

Franziska HUBERT



- **Contexte**
- **Étude bibliographique**
  - NetFPGA
  - Les Outils de développement
  - Les Algorithmes
  - NetFlow
  - Conclusions de l'étude bibliographique
- **Le plan d'avancement**

## ■ La surveillance de trafic

- Filtrage, contrôle d'accès
- Vitesse croissante
- Diversité de menaces
- → Nécessité de développement de techniques adaptées au débit du trafic.

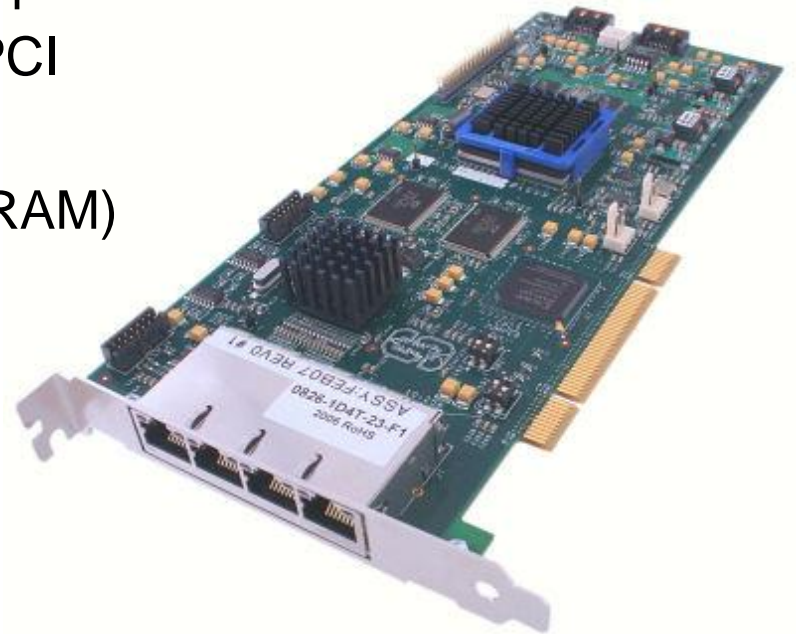
## ■ Intégrer en FPGA un algorithme de surveillance

- → Ex. classification/heavy hitters, contrôle d'accès
- Performances : complexité et débit d'une intégration matérielle
- → Comparaison de performances avec approche logicielle
- Choix du NetFPGA

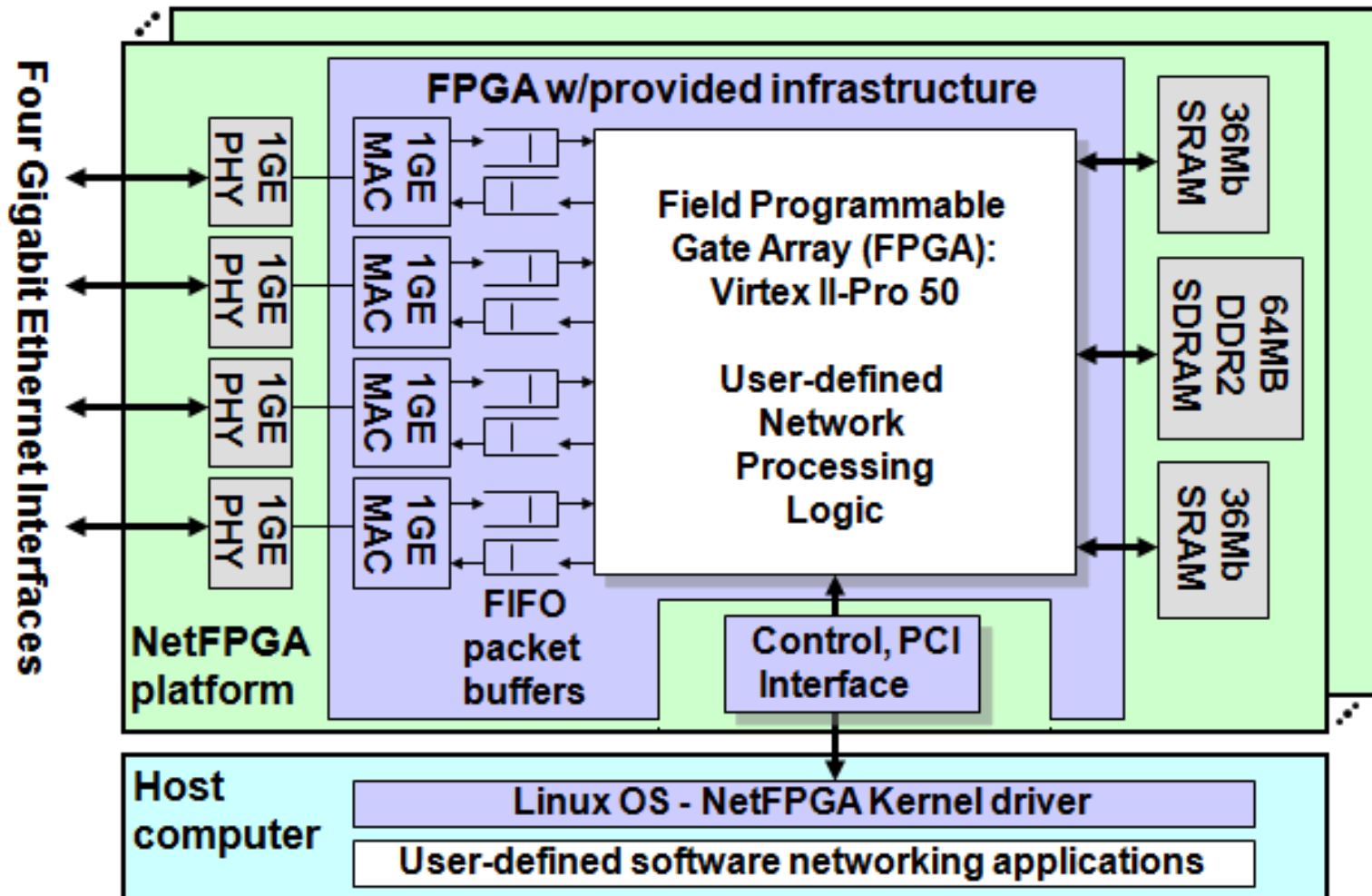
- **Contexte**
- **Étude bibliographique**
  - **NetFPGA**
  - Les Outils de développement
  - Les Algorithmes
  - NetFlow
  - Conclusions de l'étude bibliographique
- **Le plan d'avancement**

## Carte PCI embarquant:

- un FPGA Xilinx **Virtex2-Pro 50**: logique 'utilisateur'
- un FPGA Xilinx Spartan II: logique PCI
- 4 ports **Gigabit Ethernet**
- 4.5MB (2x18Mb) de SRAM (Static RAM)
- 64 MB de DDR2 DRAM
- 2 ports **SATA**



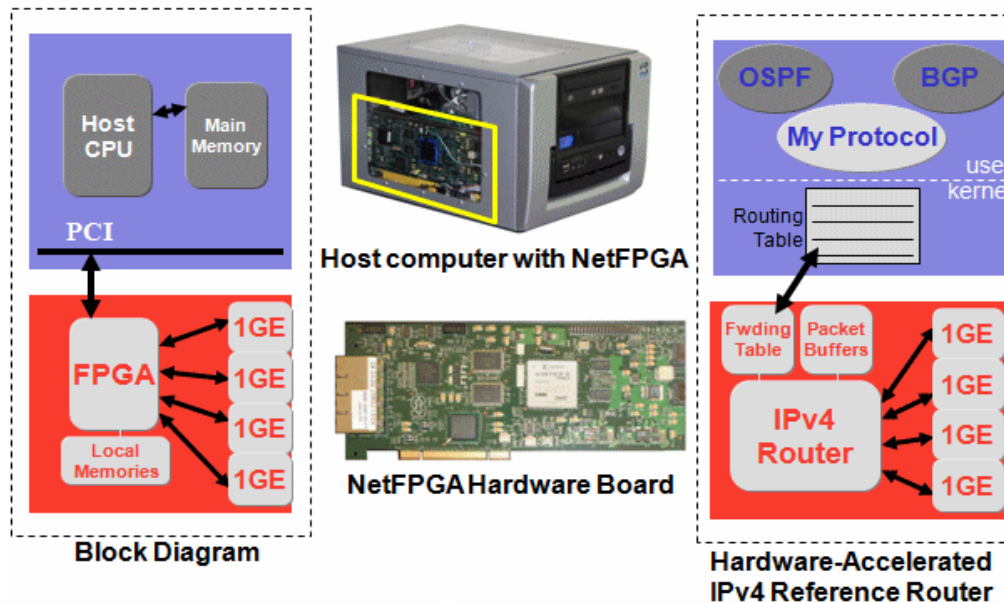
# NetFPGA : Architecture



# NetFPGA : Intérêt

- Connecté à un PC
- Décharge le CPU de celui-ci du traitement des données
- CPU:
  - accès à la mémoire principale
  - peut effectuer des opérations DMA de lecture/d'écriture

**Accélération matérielle**



**NFPs** (Paquets NetFPGA) : code implémentant des fonctions réseaux

**3 façons** de procéder en utilisant le NFP « *reference router* » comme base:

- Configurer le FPGA de la carte en routeur et ensuite modifier la partie logicielle (programme utilisateur, pilote Linux).
- Utiliser le code FPGA du routeur et étendre ses fonctionnalités en développant un module utilisateur.
- Créer un design FPGA totalement nouveau.

- **Contexte**
- **Étude bibliographique**
  - NetFPGA
  - **Les Outils de développement**
  - Les Algorithmes
  - NetFlow
  - Conclusions de l'étude bibliographique
- **Le plan d'avancement**



# Xilinx, ModelSim, Chipscope

- **Xilinx ISE**

Contrôle de flux de conception :

- Concevoir
- Simuler
- Exécuter
- Simulation temporellement adaptée

- **ModelSim**

- Simulation pour des HDL (Hardware Description Language)
- Environnement de simulation
- Debogage VHDL, Verilog, SystemC
- Modèles numériques et analogiques
- Simulation temporellement adaptée

- **Chipscope**

- Analyse des signaux sur le FPGA
- → Ajout de la logique nécessaire
- Fonctionne à partir d'une exécution réelle sur le FPGA



# VHDL, Verilog

- Décrit le matériel à un niveau d'abstraction élevé
- Décrit les circuits en définissant leurs modules, portes logiques et expressions
- Combinaison des éléments décrits rend possible la synthèse du circuit et son test

- Plateforme pour réaliser le prototypage de MP-SoC
- Fournis des outils d'exploration d'architecture
- Cœur: bibliothèque de composants virtuels
- Accélération de simulation : modèles automatiquement générés
- Évaluation de performance plus tôt dans le processus de conception
- Problème : ne génère pas de code Verilog  
→ SC2V ?
- **Choix:**
  - SoCLib : Contraintes de SC2V ou traduction à la main
  - Écrire directement en Verilog



# Plan

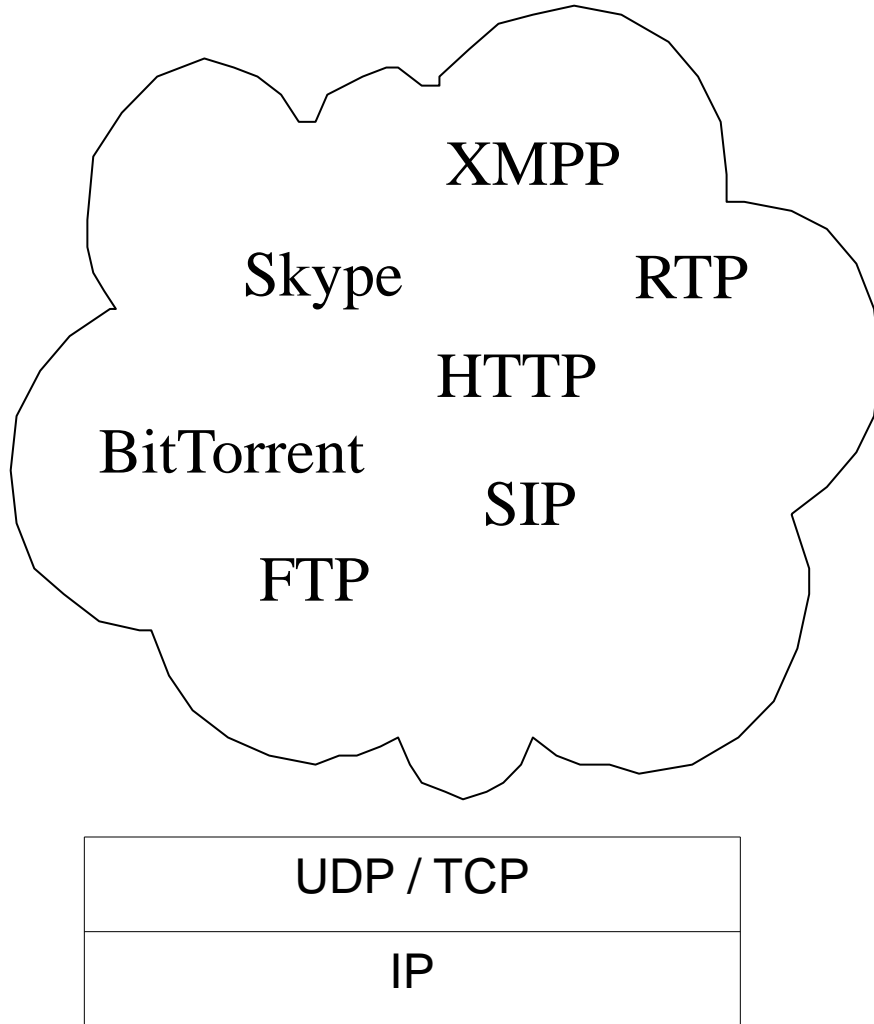
- **Contexte**
- **Étude bibliographique**
  - NetFPGA
  - Les Outils de développement
  - **Les Algorithmes**
  - NetFlow
  - Conclusions de l'étude bibliographique
- **Le plan d'avancement**



# Les algorithmes

- **Problématiques réseau**
  - La classification de services
  - Les attaques
- **Stream Mining**
- **L'algorithme CMS**
- **La mémoire Trie**
- **Nos choix**

# Classification de services



## Utilisation :

- la QoS
- la facturation

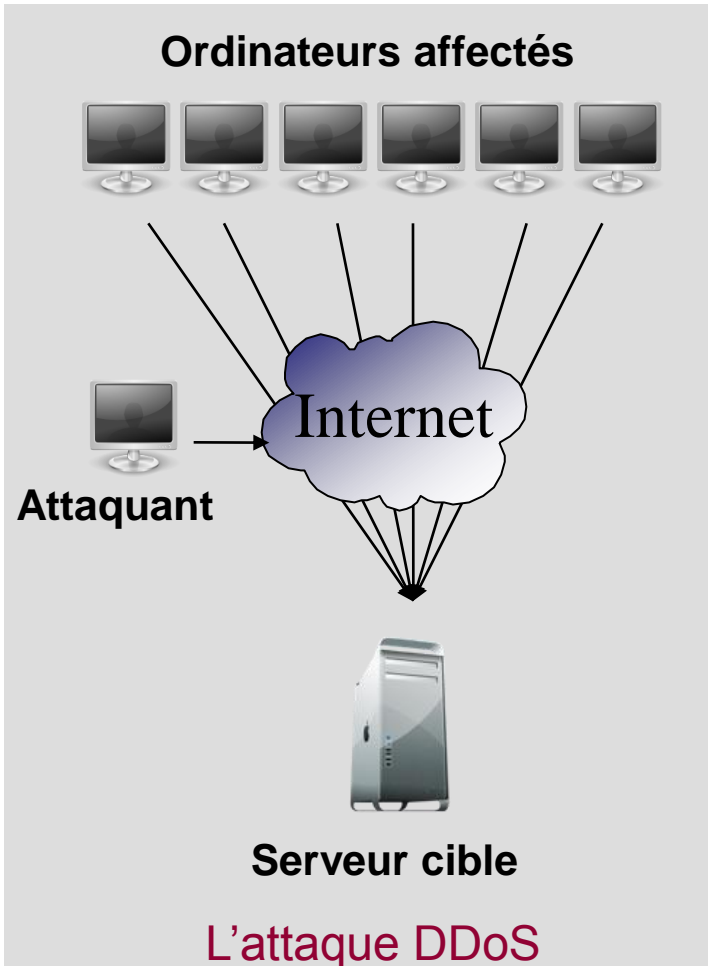
## Analyse :

- ports UDP/TCP
- répartition des paquets

## Reconstitution des flux :

- mémoire importante
- timers

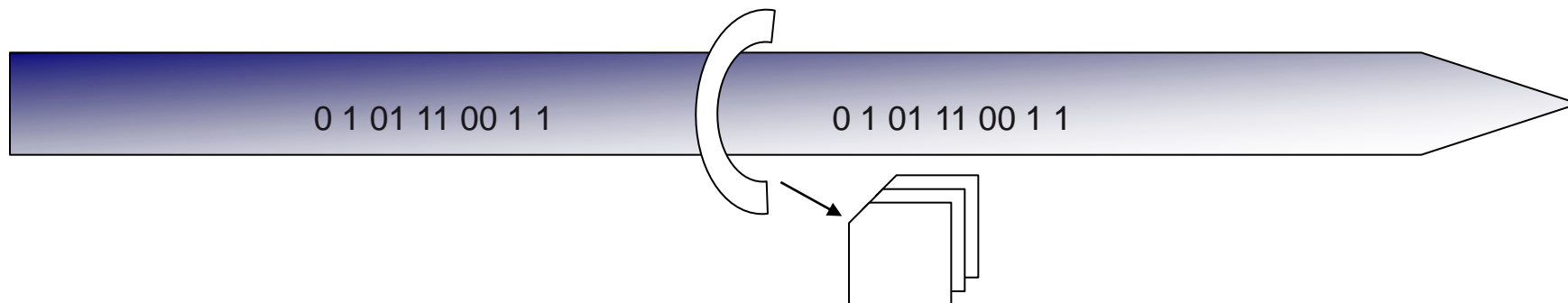
# Attaques



## Attaques:

- DoS/DDoS : (Distributed) Denial of Service
  - SYN Flooding
  - PING Flooding
  - TearDrop
- Interception de communication (Man in the middle)
- Trappes (Troyens)

# Le Stream Mining



## Analyser un flux de données :

- En temps réel
- Sans stocker le flux

## → Résultat approché :

- Déterministe
- Probabiliste

## Comptage des paquets SYN envoyés par chaque IP :

- 100Go/s pendant 5 min => 30To
- 4 294 967 296 adresses IPv4

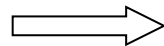
# L'algorithme CMS

**CMS : Count Min. Sketch** : Cormode et Muthukrishnan en 2004

- Algorithme de Stream Mining
- *objectif* : compter efficacement les paquets IP envoyés par chaque source

**Principe :**

- Précision  $\epsilon$
- Probabilité d'erreur  $\delta$



- $d = \ln(1/\delta)$  fonctions de hachage
- $w = e/\epsilon$  valeurs d'arrivée

$\varphi : \text{IP source} \longrightarrow 1 \dots w$

# L'algorithme CMS - exemple

	$\varphi_1$	$\varphi_2$	$\varphi_3$	$\varphi_4$	$\varphi_5$
<b>1</b> = $\varphi_3(\text{IP}_0)$	0	0	<b>1</b>	0	0
<b>2</b> = $\varphi_1(\text{IP}_0)$	<b>1</b>	0	0	0	0
<b>3</b>	0	0	0	0	0
<b>4</b> = $\varphi_2(\text{IP}_0) = \varphi_5(\text{IP}_0)$	0	<b>1</b>	0	0	<b>1</b>
<b>5</b>	0	0	0	0	0
<b>6</b> = $\varphi_4(\text{IP}_0)$	0	0	0	<b>1</b>	0

# L'algorithme CMS - exemple

	$\varphi_1$	$\varphi_2$	$\varphi_3$	$\varphi_4$	$\varphi_5$
<b>1</b> = $\varphi_4(\text{IP1})$	0	0	1	<b>1</b>	0
<b>2</b> = $\varphi_5(\text{IP1})$	1	0	0	0	<b>1</b>
<b>3</b> = $\varphi_1(\text{IP1})$	<b>1</b>	0	0	0	0
<b>4</b> = $\varphi_2(\text{IP1})$	0	<b>2</b>	0	0	1
<b>5</b>	0	0	0	0	0
<b>6</b> = $\varphi_3(\text{IP1})$	0	0	<b>1</b>	1	0

# L'algorithme CMS - exemple

	$\varphi_1$	$\varphi_2$	$\varphi_3$	$\varphi_4$	$\varphi_5$
<b>1</b> = $\varphi_4(\text{IP1})$	0	0	1	<b>2</b>	0
<b>2</b> = $\varphi_5(\text{IP1})$	1	0	0	0	<b>2</b>
<b>3</b> = $\varphi_1(\text{IP1})$	<b>2</b>	0	0	0	0
<b>4</b> = $\varphi_2(\text{IP1})$	0	<b>3</b>	0	0	1
<b>5</b>	0	0	0	0	0
<b>6</b> = $\varphi_3(\text{IP1})$	0	0	<b>2</b>	1	0

# L'algorithme CMS

## Certitudes :

$$P(\hat{a} \leq a + \varepsilon N) \geq 1 - \delta$$
$$\hat{a} \geq a$$

- Algorithme de Stream Mining
- $a$  : nombre de paquets reçus de IPx
- $\hat{a}$  : estimation de  $a$
- $N$  : somme de tous les paquets reçus

## Complexité :

- En temps :  $O(d) = O(\ln(1/\delta))$
- En espace :  $O(dw) = O(e \cdot \ln(1/\delta)/\varepsilon)$

# La mémoire Trie

- Création d'une mémoire à partir d'un arbre
- Utile pour les pare-feux et routeurs
- Intérêt potentiel pour notre projet :

Trier les paquets selon les en-têtes

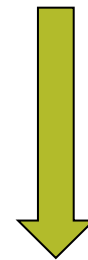


Règles

1.2.\* => forward  
\* => drop

**Temps de traitement :**

- Constant par paquet
- Indépendant du nombre de règles



Mémoire Trie

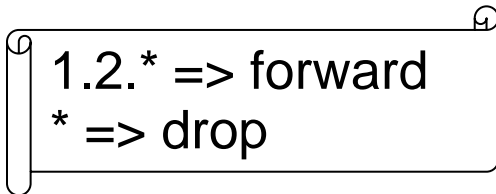


Arbre

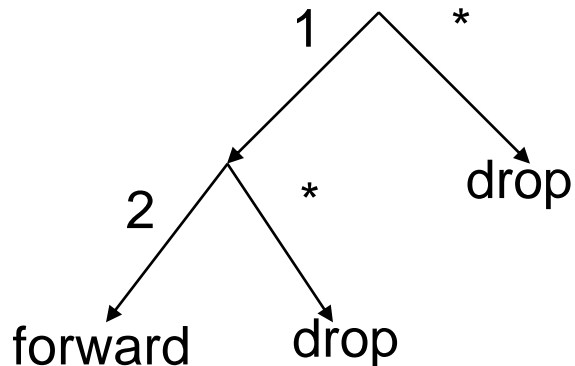
# La mémoire Trie

## ■ Exemple

### 1) Règles



### 2) Arbre



### 3) Mémoire Trie

	1	2	3	4
Règle 1	→ 2	drop	drop	drop
Règle 2	drop	forward	drop	drop

4) L'adresse 1.3.4.5 envoie un paquet

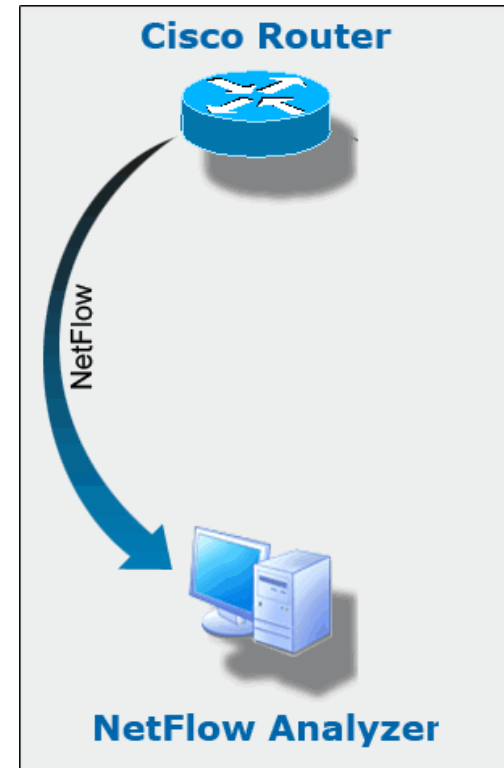
- **La problématique :**
  - Détection d'attaques par SYN flooding
- **L'outil :**
  - Algorithme CMS
- **Possibilités d'évolution :**
  - Il serait envisageable d'utiliser la mémoire Trie pour généraliser à d'autres attaques de type "flooding"



# Plan

- **Contexte**
- **Étude bibliographique**
  - NetFPGA
  - Les Outils de développement
  - Les Algorithmes
  - **NetFlow**
  - Conclusions de l'étude bibliographique
- **Le plan d'avancement**

- **Protocole** réseau propriétaire développé par **Cisco Systems** pour leurs routeurs
- Collecte des **informations de trafic IP**
- Supporté par :
  - Cisco IOS
  - Linux
  - BSD
- **Enregistrements NetFlow:**
  - générés par les routeurs
  - exportés dans des paquets UDP ou SCTP
  - collectés par un collecteur Netflow.



# NetFlow : les enregistrements (1)

## Contenu d'un enregistrement NetFlow v5:

- Numéro de version
- Numéro de séquence
- Interface d'entrée et de sortie
- Timestamp de début et de fin du flux
- Nombre d'octets et de paquets observés dans le flux

# NetFlow : les enregistrements (2)

## Contenu d'un enregistrement NetFlow v5 (suite):

- Les en-têtes niveau 3:
  - IP source et destination
  - Port source et destination
  - Numéro de protocole IP
  - Valeur du ToS (Type of Service)
- Dans le cas d'un flux TCP, l'union de tous les drapeaux TCP observés durant la vie du flux
- Les informations de routage niveau 3:
  - Adresse IP du "next hop" immédiat dans la route vers la destination
  - Masques de sous réseau des IP source et destination

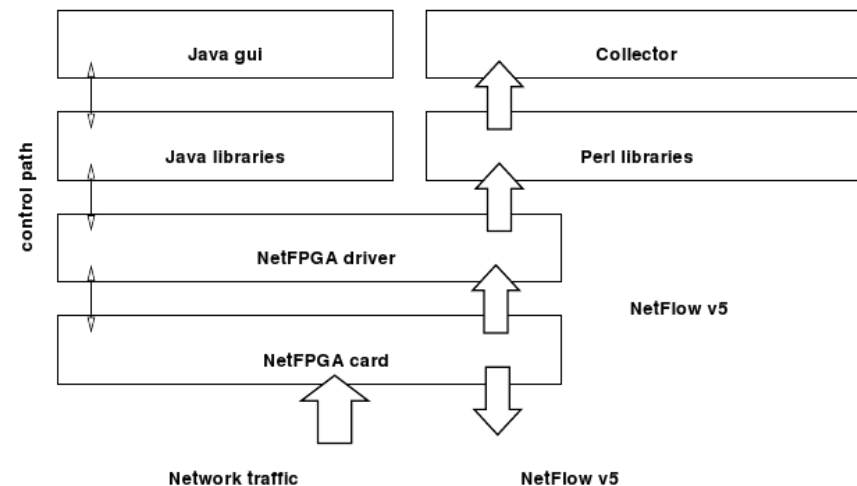
# Sonde NetFPGA NetFlow

## • Organisation:

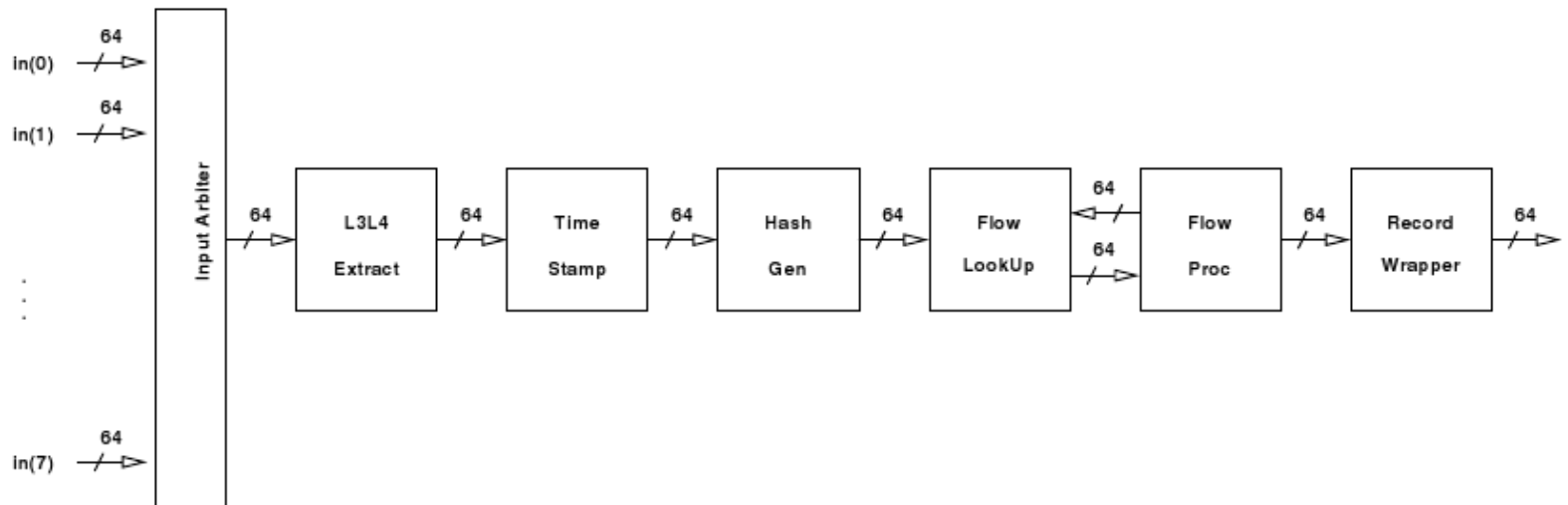
- Un ordinateur hôte: Contrôle, configuration et collecte des flux
- Une carte NetFPGA: mesure des flux

## • Principaux paramètres de la sonde:

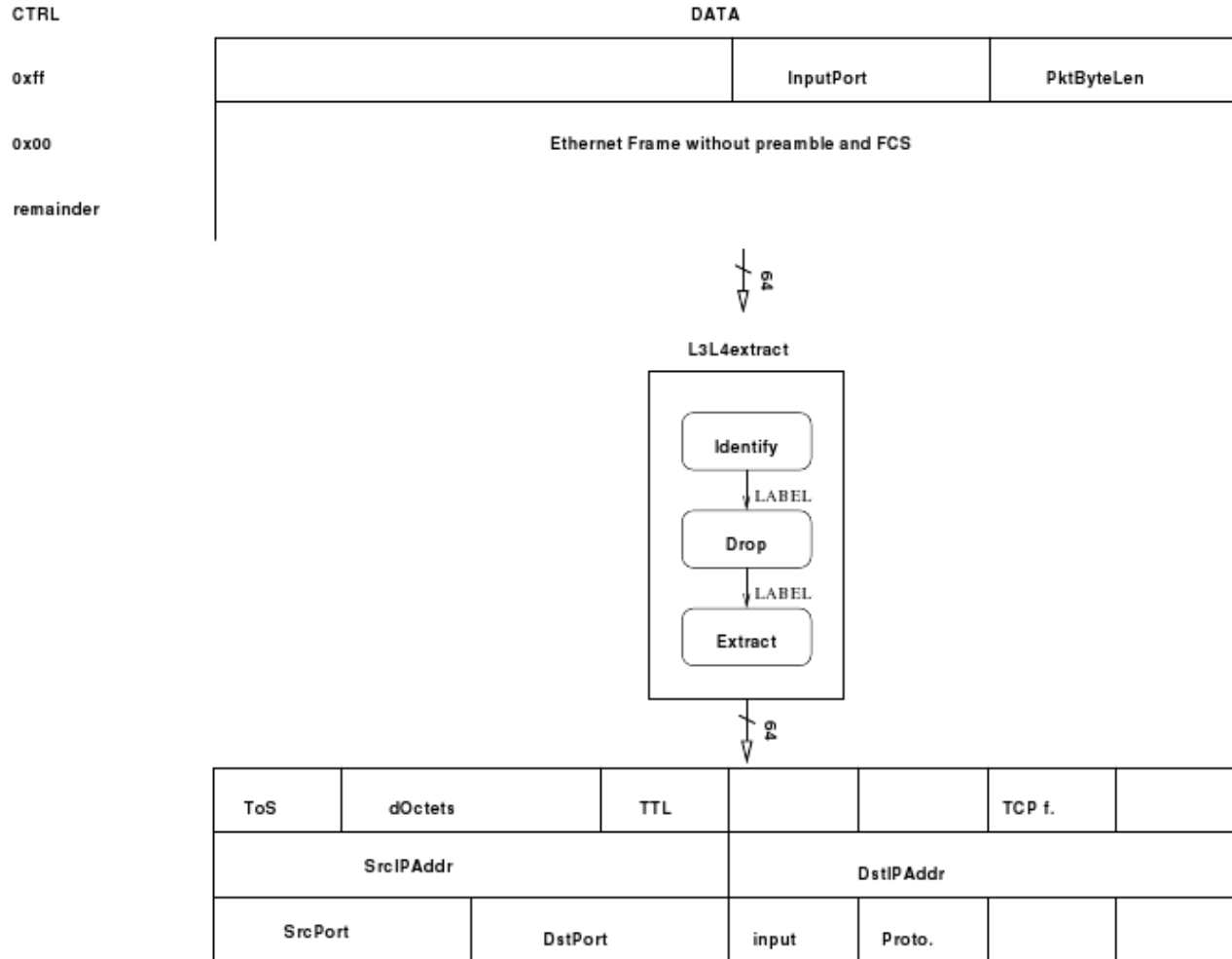
- Mesure en temps réel des flux des quatre interfaces réseau Gigabit
- Mémoire accueille 60000 flux différents
- Indexation des enregistrements de flux à l'aide d'un hash
- Export des enregistrements au format NetFlow v5



# Sonde NetFlow : architecture



# Sonde NetFlow : L3L4 parser

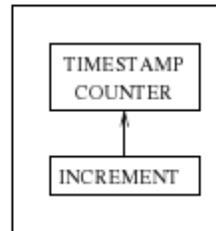


# Sonde NetFlow : Timestamp

ToS	dOctets	TTL	TCP f.			
SrcIPAddr			DstIPAddr			
SrcPort	DstPort	input	Proto.			



Timestamp



			Timestamp			
ToS	dOctets	TTL			TCP f.	
SrcIPAddr			DstIPAddr			
SrcPort	DstPort	input	Proto.			

# Sonde NetFlow : Hash

			Timestamp			
ToS	dOctets	TTL			TCP f.	
SrcIPAddr			DstIPAddr			
SrcPort	DstPort		input	Proto.		

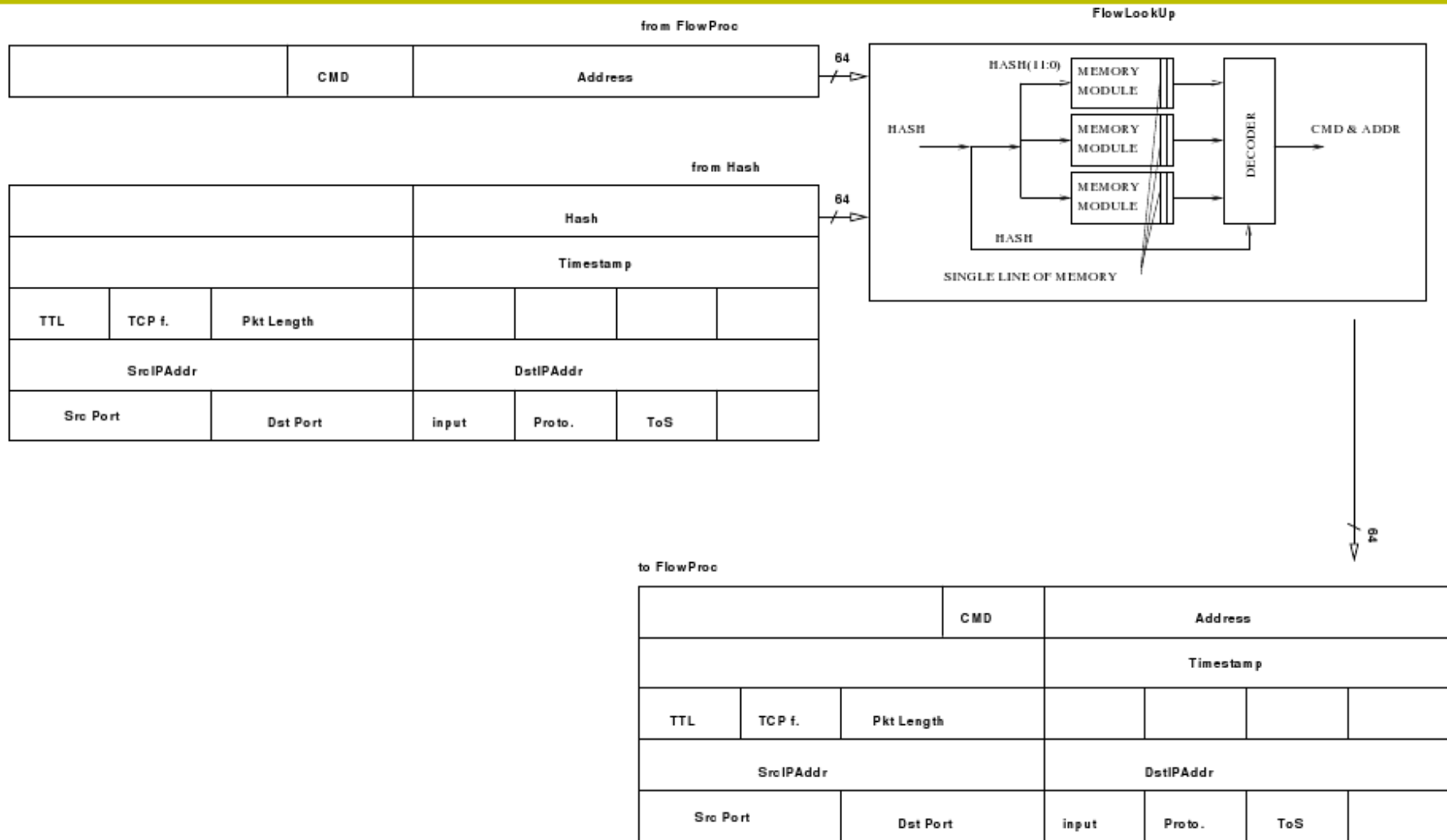
64



64

			Hash			
			Timestamp			
ToS	dOctets	TTL			TCP f.	
SrcIPAddr			DstIPAddr			
SrcPort	DstPort		input	Proto.		

# Sonde NetFlow : FlowLookUp



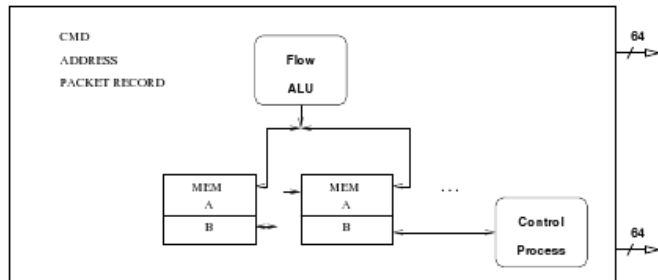
# Sonde NetFlow : FlowProc

## Contrôle:

- la **création** de nouveaux flux
- la **mise à jour** des flux existants
- l'**expiration** des flux inactifs

			Cmd	Address		
			Timestamp			
TTL	TCP f.	Pkt Length				
SrcIPAddr			DstIPAddr			
Src Port	Dst Port	input	Proto.	ToS		

FlowProc



Start Timestamp		End Timestamp			
dOctets		dPkts	TTL	TCP f.	
SrcIPAddr		DstIPAddr			
Src Port	Dst Port	input	Proto.	ToS	

			Cmd	Address		
--	--	--	-----	---------	--	--

# Résultats de l'étude bibliographique

## ■ NetFPGA :

- Accélération matérielle
- Implémentation de référence : routeur IP
- Exemple de communication ordinateur/NetFPGA : NetFlow

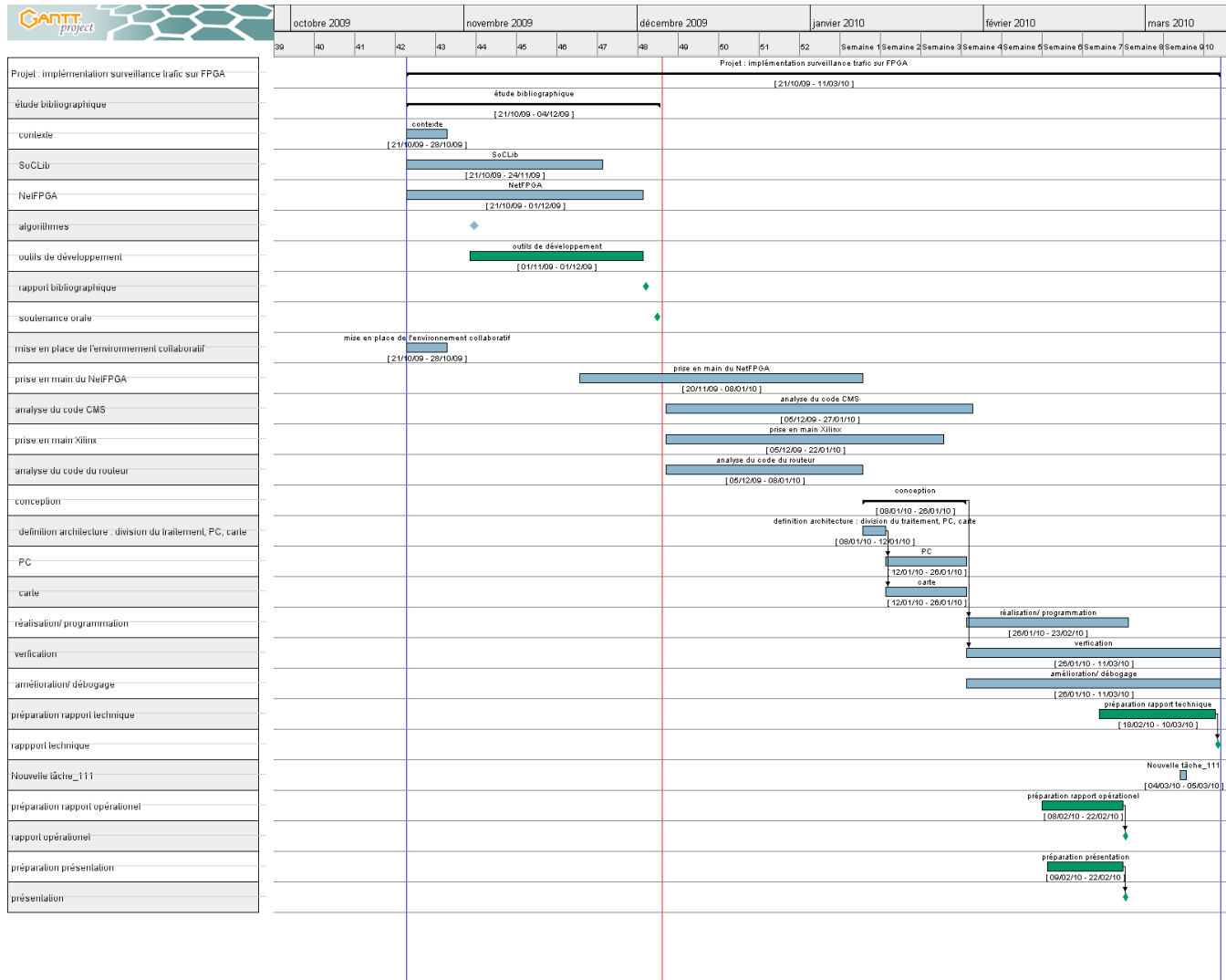
## ■ Problématique : détection d'attaques par SYN flooding

- Algorithme : CMS (recherche d'objets massifs)
- Possibilités d'évolution

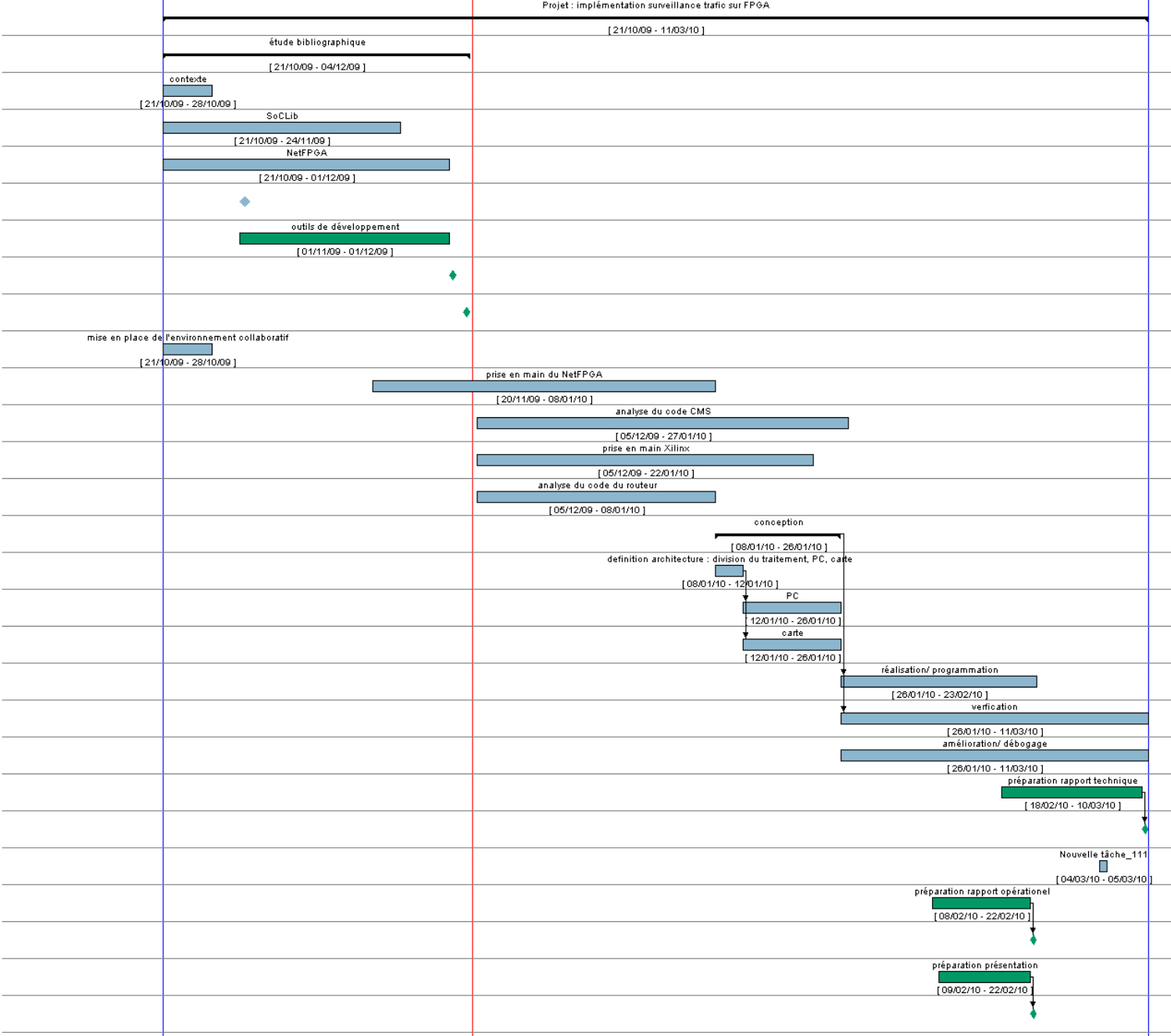
## ■ Développement :

- Directement en Verilog
- Utilisation de la suite de développement Xilinx SE

# Plan d'avancement



Projet : implémentation surveillance trafic sur FPGA
étude bibliographique
contexte
SoCLib
NetFPGA
algorithmes
outils de développement
rapport bibliographique
soutenance orale
mise en place de l'environnement collaboratif
prise en main du NetFPGA
analyse du code CMS
prise en main Xilinx
analyse du code du routeur
conception
définition architecture : division du traitement, PC, carte
PC
carte
réalisation/ programmation
verification
amélioration/ débogage
préparation rapport technique
rapport technique
Nouvelle tâche_111
préparation rapport opérationnel
rapport opérationnel
préparation présentation
présentation



# Bibliographie

- [NET08] John W. Lockwood, G. Adam Covington, Jan Kořenek et al. NetFPGA : Tutorial in Brno, Brno, Czech Republic, 5 Septembre 2008. Disponible sur [http://netfpga.org/tutorials/Brno2008/ppt/NetFPGA\\_Brno\\_2008\\_09\\_04c.pdf](http://netfpga.org/tutorials/Brno2008/ppt/NetFPGA_Brno_2008_09_04c.pdf)
- [NET09a] NetFPGA: Guide **[En ligne]**. Disponible sur <http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/Guide> (consulté le 1er décembre 2009).
- [NET09b] NetFPGA: NetFPGA Video Demonstrations **[En ligne]**. Disponible sur <http://www.netfpga.org/php/videos.php> (consulté le 1er décembre 2009).
- [VHDL] VHDL. **[En ligne]** [http://de.wikipedia.org/wiki/Very\\_High\\_Speed\\_Integrated\\_Circuit\\_Hardware\\_Description\\_Language](http://de.wikipedia.org/wiki/Very_High_Speed_Integrated_Circuit_Hardware_Description_Language) (consulté le 1 décembre 2009).
- [VERa] Verilog. **[En ligne]** <http://en.wikipedia.org/wiki/Verilog> (consulté le 1 décembre 2009).
- [VERb] Verilog. **[En ligne]** <http://fr.wikipedia.org/wiki/Verilog> (consulté le 1 décembre 2009).
- [XIL-ISE] Xilinx ISE 11. **[En ligne]** [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx11/ise11tut.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise11tut.pdf) (consulté le 30 novembre 2009).
- [MSa] ModelSim. **[En ligne]** <http://www.model.com/content/modelsim-se-high-performance-simulation-and-debug> (consulté le 1 décembre 2009).
- [MSb] ModelSim. **[En ligne]** <http://www.mikrocontroller.net/articles/ModelSim> (consulté le 1 décembre 2009).
- [LAS04] Anselmo Lastra. *Chipscope Tutorial*. **[En ligne]** [http://www.cs.unc.edu/~lastra/comp190/Notes/13\\_Chipscope.pdf](http://www.cs.unc.edu/~lastra/comp190/Notes/13_Chipscope.pdf) (consulté le 1 décembre 2009), Février 2004
- [XIL09] Xilinx, *ChipScope Pro and the Serial I/O Toolkit*. **[En ligne]** <http://www.xilinx.com/tools/cspro.htm> (consulté le 1er décembre 2009). 2009

# Bibliographie

- [SoCL08] SoCLib. **[En ligne]** Disponible sur <https://www.soclib.fr/trac/dev/wiki> (consulté le 30 novembre 2009).
- [BCC07] Tian Bu, Jin Cao, Aiyou Chen, Patrick P. C. Lee. *A Fast and Compact Method for Unveiling Significant Patterns in High Speed Networks*. IEEE. 2007.
- [BFG09] Yannick BLEUWART, Éric FAGOT, Grégory GIEMZA, Yanick PIGNOT. *Séminaire de détection d'intrusions Dos, DDos. Attaque du Syn Flooding : exemple sur un serveur web Apache*. **[En ligne]** Disponible sur <http://www.student.montefiore.ulg.ac.be/~bleuwart/> (consulté le 30 novembre 2009).
- [CHE05] Pascal CHEUNG. *Une introduction aux techniques de stream mining*. France Telecom. **[En ligne]** Disponible sur [http://trac.benoute.fr/netfpga/raw-attachment/wiki/CMS/t\\_12janvier05\\_c.pdf](http://trac.benoute.fr/netfpga/raw-attachment/wiki/CMS/t_12janvier05_c.pdf) (consulté le 30 novembre 2009), 12 janvier 2005.
- [FRE60] Edward Fredkin. *Trie Memory*. **[En ligne]** Disponible sur <http://delivery.acm.org/10.1145/370000/367400/p490-fredkin.pdf?key1=367400&key2=4663059521&coll=GUIDE&dl=GUIDE&CFID=64097606&CFTOKEN=74272576> (consulté le 30 novembre 2009), 1960.
- [GOM09] Sylvain Gombault. Octobre 2009. *Présentation du projet VTHD++*.
- [LUD05] BoonPing Lim, Md. Safi Uddin. *Statistical-based SYN-flooding Detection Using Programmable Network Processor*. IEEE. **[En ligne]** Disponible sur <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1489006&isnumber=32022> (consulté le 1er décembre 2009), juillet 2005.
- [MUT05] S. Muthu Muthukrishnan. *MassDal public code bank*. **[En ligne]** Disponible sur <http://www.cs.rutgers.edu/~muthu/massdal-code-index.html> (consulté le 30 novembre 2009), 2005.
- [SLC09] Robert Schweller, Zhichun Li, Yan Chen, Yan Gao, Ashish Gupta, Yin Zhang, Peter A. Dinda, Ming-Yang Kao, Gokhan Memik. *Reversible Sketches: Enabling Monitoring and Analysis Over High-Speed Data Streams*. IEEE. 2009.
- [TAN97] K. M. C. Tan, B. S. Collie. *Detection and Classification of TCP/IP Network Services*. **[En ligne]** IEEE. Disponible sur <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=646179&isnumber=14094> (consulté le 30 novembre 2009), 1997.
- [WIK09] Wikipédia. *Trie*. **[En ligne]** Disponible sur <http://en.wikipedia.org/wiki/Trie> (consulté le 30 novembre 2009), 29 novembre 2009.

# Merci

<http://trac.benoute.fr/netfpga/>