

# Κατανεμημένα Συστήματα I

## Μάθημα Βασικής Επιλογής, Χειμερινού Εξαμήνου Τομέας Εφαρμογών και Θεμελιώσεων

Χρήστος Κονίνης

Τρίτη, 1 Δεκεμβρίου, 2009  
Υπολογιστικό



## Επισκόπηση

Σφάλματα επικοινωνίας στο Shawn  
Run exactly the same Simulation  
Σφάλματα επικοινωνίας Example

Shawn Tags  
Shawn Tags  
Tags Example

4η Άσκηση  
findval



## Σφάλματα επικοινωνίας στο Shawn

- ▶ Μέχρι τώρα όλα τα μηνύματα που έστελνε ένας κόμβος παραδίδονταν με επιτυχία
- ▶ Τι γίνεται σε περίπτωση που θέλουμε να εξομοιώσουμε σφάλματα στην επικοινωνία;
- ▶ Πρέπει να προσθέσουμε το `random_drop_chain` στην αλυσίδα των `transmission model`
- ▶ Το `random_drop_chain` απορρίπτει μηνύματα με μια συγκεκριμένη πιθανότητα



## Σφάλματα επικοινωνίας στο Shawn

```
prepare_world edge_model=list comm_model=disk_graph \  
  transm_model=stats_chain \  
  range=1  
chain_transm_model name=random_drop_chain probability=0.1  
chain_transm_model name=reliable  
  
rect_world width=50 height=50 count=5000 processors=helloworld  
simulation max_iterations=10  
  
dump_transmission_stats
```

- ▶ Το παραπάνω παράδειγμα απορρίπτει μηνύματα με πιθανότητα 10%
- ▶ Τα μηνύματα που δεν απορρίπτονται, παραδίδονται στον τελικό προορισμό από το `reliable model`
- ▶ Η αλυσίδα των `transmission models` που προκύπτει είναι :  
Message → StatsChain → RandomDropChain → Reliable



## Επαναλαμβάνοντας την ίδια εξομίωση

- ▶ Το προηγούμενο παράδειγμα παρατηρούμε ότι σε κάθε εξομίωση παίρνουμε διαφορετικά αποτελέσματα
- ▶ Αυτό συμβαίνει γιατί :
  1. Το `rect_world` τοποθετεί τους κόμβους τυχαία
  2. Το `random_drop_chain` απορρίπτει τα μηνύματα τυχαία
- ▶ Αν θέλουμε να επαναλαμβάνουμε την ίδια εξομίωση πρέπει να αρχικοποιούμε την γεννήτρια τυχαίων αριθμών με το ίδιο `seed`
- ▶ Υπάρχουν 2 τρόποι για να την αρχικοποιήσουμε :

1. Να θέσουμε απευθείας τιμή στο `seed`

```
random_seed action=set seed=123456789
```

2. Να σώσουμε το `seed` από μια εξομίωση και έπειτα να την χρησιμοποιήσουμε σε μια άλλη

```
random_seed action=create filename=file_containing_the_seed
random_seed action=load filename=file_containing_the_seed
```



## Σφάλματα επικοινωνίας Example

- ▶ Κατεβάστε το κώδικα του παραδείγματος από εδώ :  
`ftp://carrot.cti.gr/lab05/simpleconsensus.tar.gz`
- ▶ Μεταφέρεται τα αρχεία `msgerrors.conf` και `input-topology.xml` στο φάκελο `shawn/buildfiles`
- ▶ Δημιουργήστε ένα νέο φάκελο `shawn/src/legascyapps/simpleconsensus`
- ▶ Μεταφέρεται υπόλοιπα αρχεία στον φάκελο `shawn/src/legascyapps/simpleconsensus`
- ▶ Από την κονσόλα μεταβείτε στον φάκελο `shawn/buildfiles`
- ▶ Δώστε την εντολή `csmake ..src` και μετά :
  - ▶ `c` , για να ενημερωθεί για τον νέο κώδικα που προσθέσατε
  - ▶ Ενεργοποιήστε την επιλογή `simpleconsensus` (πατώντας `enter`)
  - ▶ `c` , για να ενημερωθεί για τις αλλαγές
  - ▶ `g` , για να αποθηκεύσει τις αλλαγές
- ▶ Τέλος δώστε την εντολή `make` για να γίνει `compile` το `shawn`



## Σφάλματα επικοινωνίας Example

- ▶ Ο συγκεκριμένος `Processor` υλοποιεί τον αλγόριθμο `simple consensus`
- ▶ Όλες οι διεργασίες ξεκινάνε με μία τιμή (την μεταβλητή `input_value_0`)
- ▶ Έχουν ένα πίνακα που αποθηκεύουν όλες τις τιμές που έχουν λάβει από άλλες διεργασίες (σε κάθε θέση `i` την τιμή της διεργασίας `i`)
- ▶ Στην αρχή κάθε θέσης του πίνακα αρχικοποιείται με `UNKNOWN_VALUE`
- ▶ Σε κάθε γυρο οι διεργασίες στέλνουν ένα μήνυμα με τον πίνακα στους γείτονες
- ▶ Κάθε διεργασία που λάβει ένα μήνυμα ενημερώνει τον πίνακα που έχει με τις νέες τιμές που έμαθε
- ▶ Οι διεργασίες τερματίζουν μετά από `diameter` γύρους και εξετάζοντας τις τιμές του πίνακα αποφασίζουν για μία τιμή (μεταβλητή `output_value_0`)



## Σφάλματα επικοινωνίας Example

- ▶ Εκτελέστε τον `simpleconsensus` χρησιμοποιώντας το `msgerrors.conf`
- ▶ Παρατηρούμε ότι όλες οι διεργασίες αποφασίζουν την ίδια τιμή  

```
----- BEGIN ITERATION 4
processors.simpleconsensus: Output value : max value : 42 majority : 42
processors.simpleconsensus: Output value : max value : 42 majority : 42
processors.simpleconsensus: Output value : max value : 42 majority : 42
processors.simpleconsensus: Output value : max value : 42 majority : 42
...
----- DONE ITERATION 4
( 0 active , 0 sleeping , 7 inactive )
```
- ▶ Τροποποιήστε την πιθανότητα στο `random_drop_chain` του `msgerrors.conf` και επαναλάβετε την εξομίωση
- ▶ Θέστε την πιθανότητα ίση με 0.9 (90%) , πόσους γύρους χρειάζεται για να αποφασίσουν την ίδια τιμή όλες οι διεργασίες :



Σφάλματα επικοινωνίας στο Shawn  
Σφάλματα επικοινωνίας στο Shawn  
Rerun exactly the same Simulation  
Σφάλματα επικοινωνίας Example

Shawn Tags  
Shawn Tags  
Tags Example

4η Άσκηση  
findval



- ▶ Μέχρι τώρα χρησιμοποιούσαμε `processors` για να υλοποιήσουμε έναν αλγόριθμο
- ▶ Κάθε `processor` είχε την δικιά του εσωτερική κατάσταση (μεταβλητές που ήταν ιδιωτικές).
- ▶ Ο μόνος τρόπος για έχουμε πρόσβαση στην εσωτερική κατάσταση του `processor` ήταν μέσω `debug` μηνυμάτων
- ▶ Όμως σε πολλές περιπτώσεις θέλουμε να αποθηκεύσουμε την κατάσταση ενός κόμβου
- ▶ Επιπλέον σε αρκετές περιπτώσεις αλγόριθμοι χρειάζονται τιμές σαν είσοδο που παράγονται από άλλους πολύπλοκους αλγόριθμους
- ▶ Τα `Tags` είναι ένας μηχανισμός του `shawn` για να προσθέτουμε πληροφορία (μεταβλητές) στους κόμβους
- ▶ Χρησιμοποιώντας τα `tasks load/save world` μπορούμε να αποθηκεύουμε και να ανακτούμε αυτές τις πληροφορίες



- ▶ Πρακτικά τα `Tags` είναι κλάσεις που περιέχουν μία τιμή (π.χ. μια μεταβλητή) και ένα όνομα (`string`)
- ▶ Υπάρχουν 3 διαφορετικές κατηγορίες `Tags` :
  1. **Simple Tags**  
Περιέχουν τιμές ενός συγκεκριμένου τύπου (π.χ. `boolean`, `int`, `string`)
  2. **Group Tags**  
Περιέχουν άλλα `Tags`
  3. **Map Tags**  
Περιέχουν ζεύγη τιμών συγκεκριμένου τύπου



1. Ορίζουμε μια μεταβλητή δείκτη τύπου `shawn::<type>Tag` στο `processor.h` π.χ.

```
shawn::IntegerTag *Int_Tag;
```
2. Την αρχικοποιούμε με ένα νέο αντικείμενο `shawn::<type>Tag` συνήθως στην `boot()` του `processor.cpp` π.χ.

```
Int_Tag = new shawn::IntegerTag( "tag_name", value );
```
3. Προαιρετικά μπορούμε να θέσουμε το `Tag` μόνιμο (`set_persistency(true)`) ώστε να αποθηκεύετε αυτόματα με το `save_world` π.χ.

```
Int_Tag->set_persistency(true);
```
4. Προσθέτουμε το νέο `Tag` στον κόμβο με την συνάρτηση `owner_w().add_tag()`; π.χ.

```
owner_w().add_tag( Int_Tag );
```



## Πώς φτιάχνουμε ένα νέο Tag

### ▶ Παραδείγματα αρχικοποίησης διαφορετικών **Simple Tags**

#### ▶ **Integer Tag :**

```
shawn::IntegerTag <int_tag;  
int_tag = new shawn::IntegerTag( "tag_name", value );  
int_tag->set_persistence(true);  
owner_w().add_tag( int_tag );
```

#### ▶ **Boolean Tag :**

```
shawn::BoolTag <bool_tag;  
bool_tag = new shawn::BoolTag( "tag_name", value );  
owner_w().add_tag( bool_tag );
```

#### ▶ **String Tag :**

```
shawn::StringTag <string_tag;  
string_tag = new shawn::StringTag( "tag_name", value );  
owner_w().add_tag( string_tag );
```



## Πώς διαβάζουμε/θέτουμε τιμή σε ένα Tag

- ▶ Στην περίπτωση που έχουμε ένα δείκτη στην μεταβλητή Tag :
  - ▶ Καλούμε την μέθοδο **set\_value()** για να θέσουμε μια νέα τιμή
  - ▶ Καλούμε την μέθοδο **value()** για να πάρουμε την τιμή

### Παράδειγμα χρήσης Tag

```
int_tag->set_value(42);  
int int_value = int_tag->value()  
  
string_tag->set_value("test");  
String string_value = string_tag->value()  
  
bool_tag->set_value(true);  
bool bool_value = bool_tag->value()
```



## Πώς βρίσκουμε ένα υπάρχων Tag από το αρχείο τοπολογίας

- ▶ Όταν υπάρχουν Tag στο αρχείο τοπολογίας (π.χ topology.xml) τότε φορτώνονται από τον εξομοιωτή πριν ξεκινήσει η εξομίσωση
- ▶ Σε αυτή την περίπτωση ξέρουμε μόνο το όνομα του Tag και θέλουμε να πάρουμε την τιμή του:

- ▶ Καλούμε την μέθοδο **owner\_w().find\_tag\_w("tag name")** για να πάρουμε το TagHandle που περιέχει το Tag

```
shawn::TagHandle tag = owner_w().find_tag_w("parent");
```

- ▶ Καλούμε την μέθοδο **tag.get()** και κάνουμε **dynamic\_cast** για να πάρουμε το πραγματικό Tag

```
shawn::StringTag <parentTag =  
dynamic_cast<shawn::StringTag*>( tag.get() );
```

- ▶ Καλούμε την μέθοδο **set\_value(), value()** για να πάρουμε/θέσουμε τιμή

```
string_tag->set_value("v0");  
string string_value = int_tag->value()
```



## Προσέλαση των Tags από το αρχείο τοπολογίας

### Παράδειγμα προσέλασης Tag από topology.xml

```
shawn::TagHandle tag = node.find_tag_w( "input" );  
  
//check if the tag exist and has been added to the node  
if ( tag.is_not_null() )  
{  
    //cast to the specific Tag type  
    input_tag = dynamic_cast<shawn::IntegerTag*>( tag.get() );  
}  
input_tag->set_value( 42 );  
input_tag->set_persistence( true );
```



## Tags Example

- ▶ Θα τροποποιήσετε το Processor.cpp στο `shawn/src/legacyapps/simpleconsensus` ώστε να φορτώνει την μεταβλητή `input_value_` από το Tag "input\_value" του "input\_topology.xml" :

```
<snapshot id="0">
...
<node id="v0">
<location x="0" y="0.2" z="0" />
<tag type="int" name="input_value" value="41" />
</node>
<node id="v1">
<location x="0" y="1" z="0" />
<tag type="int" name="input_value" value="42" />
</node>
....
</snapshot>
```
- ▶ Θα χρησιμοποιήσουμε ένα νέο Tag με όνομα "output\_value" ώστε να αποθηκεύουμε την μεταβλητή `output_value_`
- ▶ Θα εκτελέσουμε την εξομίωση με την τοπολογία `input-topology.xml` και θα σώσουμε την εξομίωση σε ένα αρχείο `output-topology.xml`



## Tags Example

1. Δηλώστε 2 μεταβλητές τύπου `shawn::IntegerTag*` στο `Processor.h` με όνομα `input_tag_` `output_tag_`

```
shawn::IntegerTag *input_tag_;
shawn::IntegerTag *output_tag_;
```

2. Αρχικοποιήστε τις μεταβλητές στην **boot()**

```
output_tag_ = new shawn::IntegerTag( "output_value", output_value_,
owner_w(), add_tag( output_tag_ );
output_tag_->set_persistence(true);
```

```
shawn::TagHandle tag = owner_w().find_tag_w( "input_value" );
if ( tag.is_not_null() )
{
    input_tag_ = dynamic_cast<shawn::IntegerTag*>( tag.get() );
}
input_value_ = input_tag_->value();
```

3. Μετά στην συνάρτηση **work()** βρίσκουμε το σημείο που ενημερώνουμε την μεταβλητή `output_value_` και ενημερώνουμε το αντίστοιχο Tag

```
output_tag_->set_value( output_value_ );
```



## Tags Example

- ▶ Χρησιμοποιώντας τα παρακάτω `conf` αρχείο εκτελέστε μια εξομίωση

```
prepare_world edge_model=simple comm_model=disk_graph range=1
load_world file=input-topology.xml processors=simpleconsensus
diameter=4
```

```
simulation max_iterations=100
```

```
save_world file=output-topology.xml
```

- ▶ Παρατηρήστε τα περιεχόμενα `output-topology.xml` :

```
<snapshot id="0">
...
<node id="v0">
<location x="0" y="0.2" z="0" />
<tag type="int" name="input_value" value="41" />
<tag type="int" name="output_value" value="42" />
</node>
....
</snapshot>
```



## Επισκόπηση

Σφάλματα επικοινωνίας στο Shawn

Σφάλματα επικοινωνίας στο Shawn

Rerun exactly the same Simulation

Σφάλματα επικοινωνίας Example

Shawn Tags

Shawn Tags

Tags Example

4η Άσκηση

findval



## findval - Υλοποίηση

1. Καλείστε να υλοποιήσετε έναν αλγόριθμο που κάθε διεργασία θα διαβάζει μια τιμή (input\_value)
  2. Θα στέλνει αυτή την τιμή σε μια ειδική διεργασία (Gateway Node)
  3. Η διεργασία στο Gateway Node θα υπολογίζει την ελάχιστη τιμή, την μέγιστη τιμή, τον μέσο όρο και το σύνολο, από τις τιμές που έχει λάβει
- ▶ Οι τιμές των διεργασιών θα περιέχονται στα αρχεία με τις τοπολογίες ως Tags με όνομα **input\_value**
  - ▶ Κάθε διεργασία (και το Gateway Node ) θα διαβάζει την τιμή της κατά την αρχικοποίηση από το αντίστοιχο Tag
  - ▶ Το Gateway Node θα επιλέγете χρησιμοποιώντας την συνάρτηση **special\_boot()**
  - ▶ Σε κάθε γύρο το Gateway Node θα τυπώνει από πόσους κόμβους έχει λάβει τιμή καθώς και το τρέχων ελάχιστο, μέγιστο, και μέσο όρο



## findval - Υλοποίηση

- ▶ Για την υλοποίηση μπορείτε να υλοποιήσετε όποιο αλγόριθμο (η συνδυασμό αλγορίθμων) θέλετε εσείς (π.χ. flooding, tree routing)
- ▶ Θα τρέξετε τον αλγόριθμό σας για 6 τοπολογίες αντίστοιχες με του floodmax της προηγούμενης άσκησης με όνομα (findval-topology-\*.xml)
- ▶ Θα πάρετε τιμες για την χρονική πολυπλοκότητα και πολυπλοκότητα επικοινωνίας του αλγορίθμου σας και θα σχολιάσετε την συμπεριφορά του.

