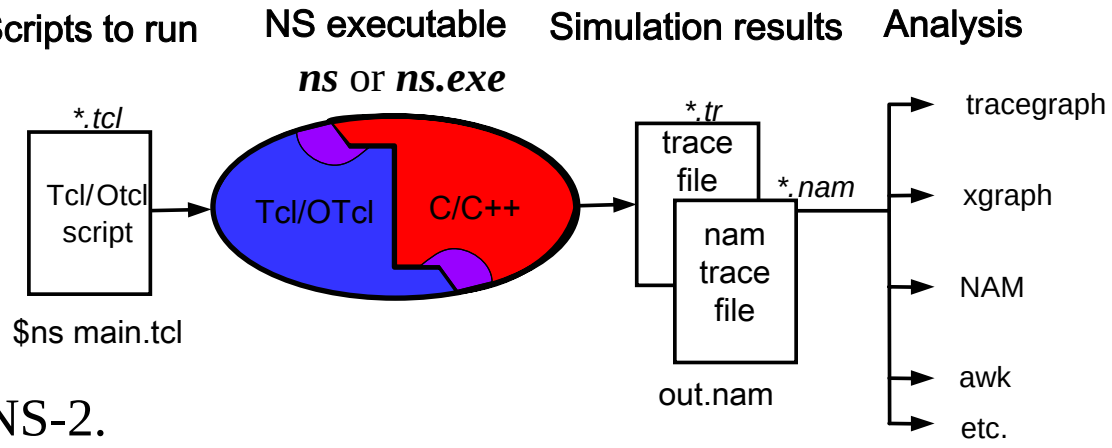
Introduction to MIT Object Tool Command Language (OTCL): Part I

Dr. Donald C. WunschII, dwunsch@umr.edu
Dr. Larry Pyeatt, pyeattl@umr.edu
Tae-hyung Kim, tk424@umr.edu

Department of Electrical Computer Engineering
University of Missouri-Rolla, USA
http://web.umr.edu/~tk424/

Milestone to teach yourself NS-2 Revisited

- **Part I**
  - Installing NS-2.
  - Running example scripts.
    "Marc Greis tutorial"
  - Learning Tcl.
  - Install tcl-debug
  - Tcl level debuggering.
- **Part II**
  - Learning OTcl.
  - C++ level debugging in NS-2.
  - OTcl/C++ binding.
  - Customize simulations with scripting (awk, bash shell).
  - C++ class hierarchy in NS-2.
  - Read the main manual for NS-2
- **Part III**
  - Write your own code. You are more than ready to go.

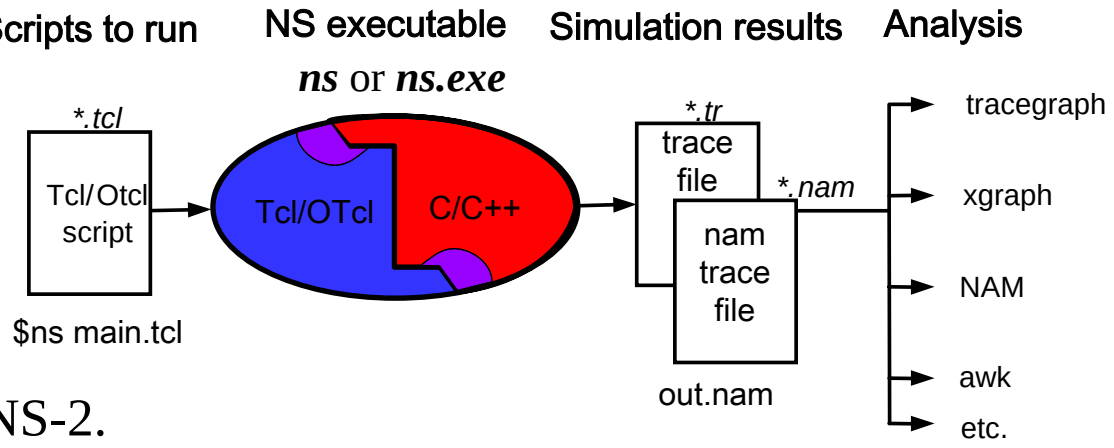


Installing tcl-debug2.0

- *tcl-debug* is a debugging tool for Tcl, made by Don Libes.
- *tcl-debug2.0* is suggested by the NS development team as the Tcl debugging tool for NS. It can be a powerful ally IF YOU KNOW HOW TO USE IT.
- Install *tcl-debug2.0*. You may refer to the tcl-debug installation manual.

Milestone to teach yourself NS-2 Revisited

- **Part I**
  - Installing NS-2.
  - Running example scripts.
    "Marc Greis tutorial"
  - Learning Tcl.
  - Install tcl-debug
  - Tcl level debuggering.
- **Part II**
  - Learning OTcl.
  - C++ level debugging in NS-2.
  - OTcl/C++ binding.
  - Customize simulations with scripting (awk, bash shell).
  - C++ class hierarchy in NS-2.
  - Read the main manual for NS-2
- **Part III**
  - Write your own code. You are more than ready to go.



Agenda

- Tcl, OTcl, and TclCL
- Comparison with C++
- Sample OTcl Scripts in NS-2
- Conclusions

Tcl, OTcl, and TclCL (1/4)

- Tcl is a string-based command language that consists of “tool commands”. Tcl has simple syntax and provides an interface to NS.
- Tcl is partly written in C language; users can write their own tool commands in C language.
- Tcl interpreter is *tclsh*, located at subdirectory *unix* in the directory for Tcl. NS release2.29 uses Tcl version 8.4.11 and the location of *tclsh* is *~/ns-allinone-2.29/tcl8.4.11/unix/tclsh*.

- OTcl is an OOP (Object-Oriented Programming) extension to Tcl/Tk, which is built on Tcl syntax and concepts. OTcl is a short for MIT Object Tcl. Relationship between Tcl and OTcl is like one between C and C++ in a sense that OTcl provides OOP to Tcl. OTcl supports classes, methods and data members of classes.
- OTcl is also written, partly, in C.
- OTcl interpreter is *otclsh*. OTcl version used by NS is 1.11. *otclsh* is located at *~/ns-allinone-2.29/otcl-1.11/otclsh.*

Tcl, OTcl, and TclCL (2/4)

- Sample OTcl commands

```
$ otclsh
% Class Fruit
% Fruit banana
% banana info class
% Fruit info instances
```

- Use *otclsh* to run an OTcl script. A sample OTcl script *firstOTclscript.otcl* is given below.

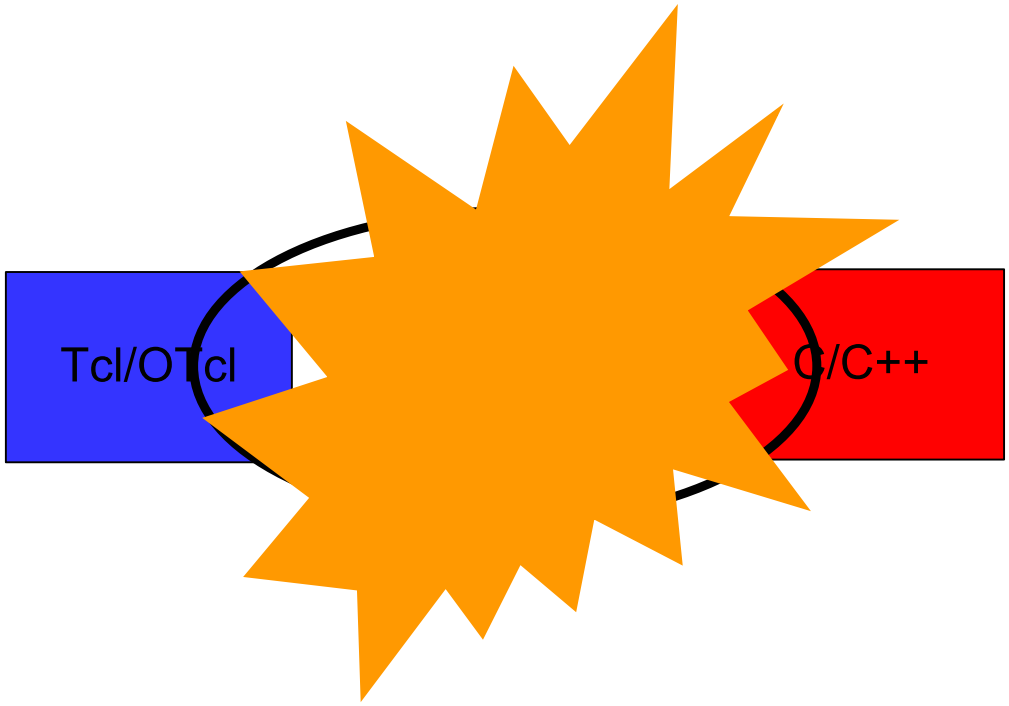
```
#!/ home/ns2userName/ns-allinone-2.29/bin/otclsh
Class Fruit
Fruit banana
set c [banana info class]
puts $c
set i [Fruit info instances]
puts $i
```

- To run *firstOTclscript.otcl*, type in

```
$ otclsh firstOTclscript.otcl
```

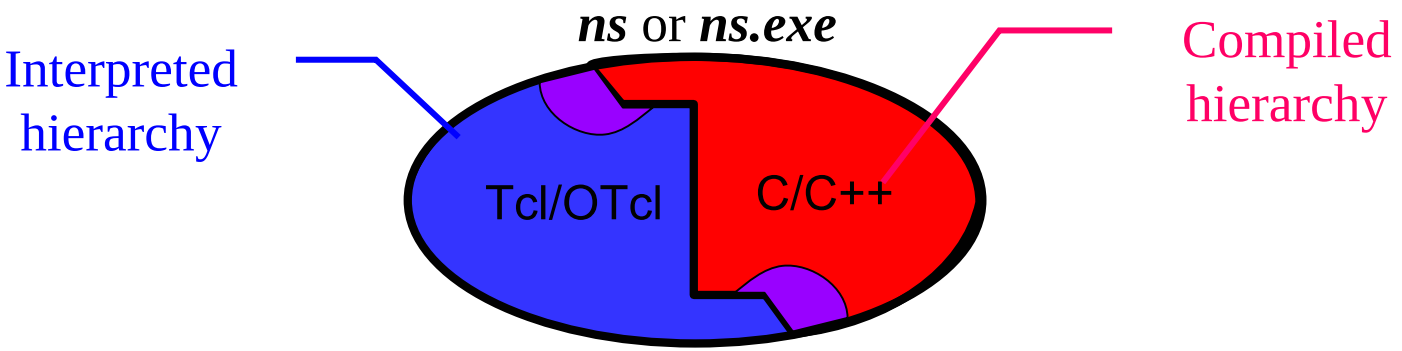
Tcl, OTcl, and TclCL (3/4)

- TclCL (Tcl with classes) is an OTcl/C++ interface, which provides a layer of C++ glue over OTcl. There is not much material explaining TclCL compared to Tcl. Fortunately, TclCL has relatively small amount of source codes, enabling to understand it by reading the source codes if necessary.
- NS-2 all-in-one package includes TclCL at *~/ns-allinone-2.29/tclcl-1.17*



Tcl, OTcl, and TclCL (3/4)

- TclCL (Tcl with classes) is an OTcl/C++ interface, which provides a layer of C++ glue over OTcl. There is not much material explaining TclCL compared to Tcl. Fortunately, TclCL has relatively small amount of source codes, enabling to understand it by reading the source codes if necessary.
- NS-2 all-in-one package includes TclCL at *~/ns-allinone-2.29/tclcl-1.17*



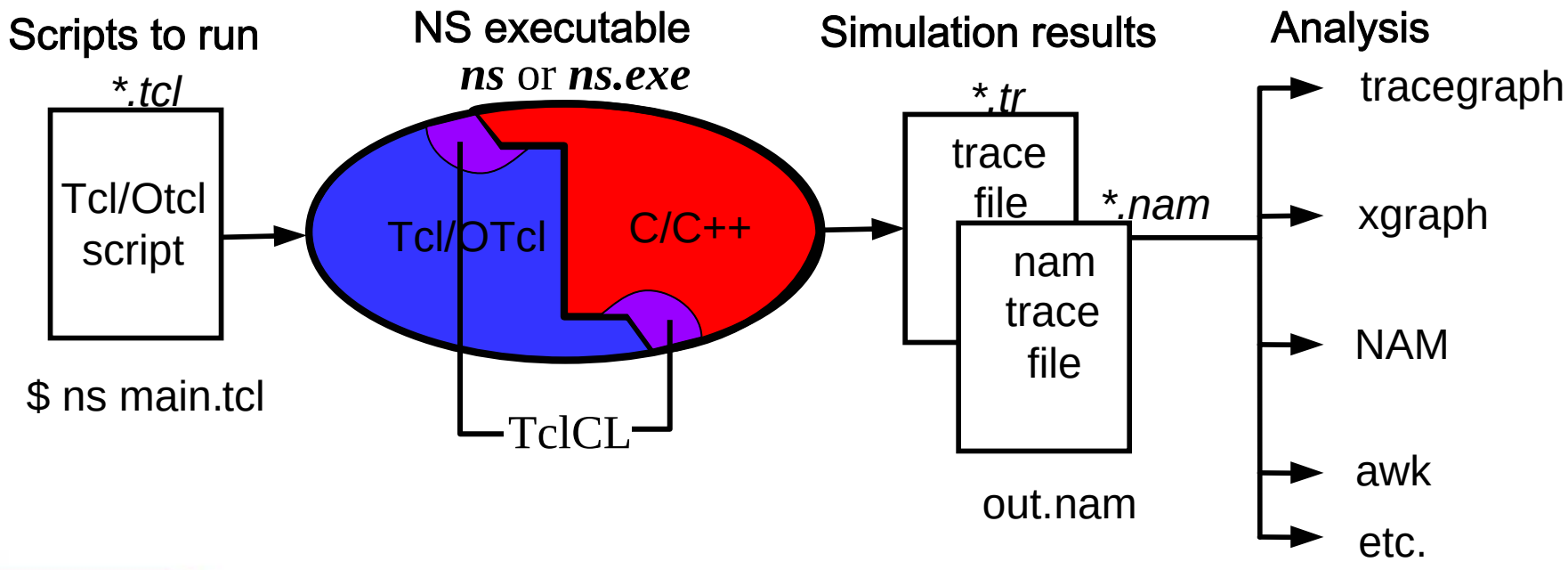
- Question: Is it possible to run Tcl/OTcl commands by using the *ns* command instead of the *tclsh* command? For example, *$ ns firstOTclScript.otcl*

Tcl, OTcl, and TclCL (4/4)

- Answer: Yes. It is because Tcl/OTcl are combined into *ns*. When you type in ns, you can see the Tcl/OTcl prompt "%". This implies Tcl/OTcl provides an interface to NS.

  $ ns
  %

- Recall the big picture,



Comparison with C++ (1/2)

• Instead of a single class declaration in C++, write multiple definitions in OTcl. Each method definition (with *instproc*) adds a method to a class. Each instance variable definition (with *set* or via *instvar* in a method body) adds an instance variable to an object.

```
% Class Fruit
% Fruit banana
% banana set flavor 0
```

• Instead of a constructor in C++, write an *init* instproc in OTcl. Instead of a destructor in C++, write a *destroy* instproc in OTcl. Unlike constructors and destructors, *init* and *destroy* methods do not combine with base classes automatically. They should be combined explicitly with *next*.

```
% Fruit instproc init {args} {
    puts "Calling Fruit instproc init"
    $self instvar color
    eval $self next $args
}
% banana destroy
```

Comparison with C++ (2/2)

• Instead of calling shadowed methods by naming the method explicitly as in C++, call them with *next*. *next* searches further up the inheritance graph to find shadowed methods automatically. It allows methods to be combined without naming dependencies.
• Unlike C++, OTcl methods are always called through the object. The name *self*, which is equivalent to this in C++, may be used inside method bodies. Unlike C++, OTcl methods are always virtual.

Sample OTcl Scripts in NS (1/2)

- Write a simple Tcl script *ex_1.tcl* with the following line.

  set fruitID [new Fruit]

- We will see an error message when we run this Tcl script by typing,

```
$ ns ex_1.tcl
invalid command name "Fruit"
    while executing
"Fruit create _o3 "
    invoked from within
"catch "$className create $o $args" msg"
    invoked from within
"if [catch "$className create $o $args" msg] {
if [string match "__FAILED_SHADOW_OBJECT_" $msg] {
delete $o
return ""
}
global errorInfo
error "class $..."
    (procedure "new" line 3)
    invoked from within
```

Sample OTcl Scripts in NS (2/2)

```
"new Fruit"
    invoked from within
"set fruitID [new Fruit]"
    (file "ex_1.tcl" line 1)
$
```

• Question: What's wrong? "*set fruitID [new Fruit]*" is perfectly valid in terms of the syntax, like "*set ns [new Simulator]*"?

• Answer: The answer will be given at the next lecture. You may spend some time to find the answer. You may com up with more questions if you think of this answer.

Conclusions

- We installed a debugger for Tcl, *tcl-debug2.0*.
- Tcl is a string-based command language that consists of “tool commands”. Tcl has simple syntax and provides an interface to NS.
- OTcl is an OOP (Object-Oriented Programming) extension to Tcl/Tk, which is built on Tcl syntax and concepts. OTcl interpreter is *otclsh*.
- TclCL (Tcl with classes) is an OTcl/C++ interface, which provides a layer of C++ glue over OTcl.
- Tcl/OTcl provides an interface to NS

```
$ ns
%
```

- Sample OTcl script.

```
Class Fruit
Fruit instproc init {args} {
    puts “Calling Fruit instproc init”
    $self instvar color
    eval $self next $args
}
```

The Next Class

- The next topic is “Introduction to OTcl (Object Tool Command Language): Part I
  - Command in OTcl

Thank you.