Washington University in St. Louis

Engineering

Forest – an Overlay Network for Real-time Distributed Apps

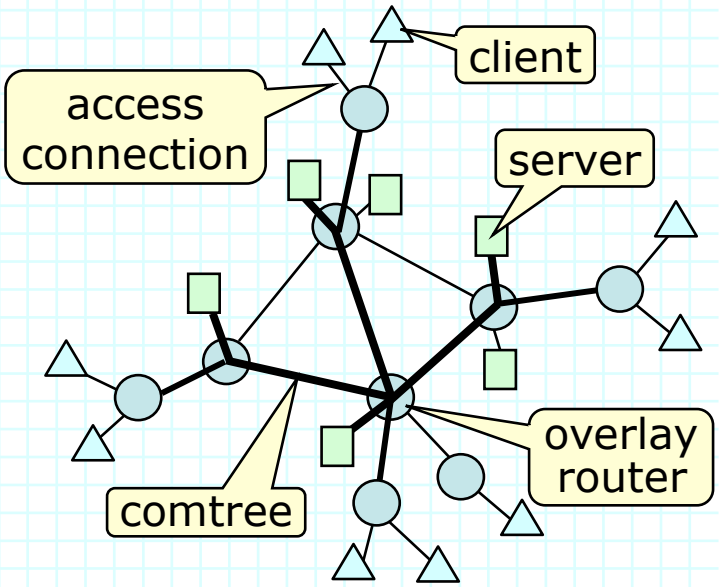
GEC7 demo – 3/2010

Jon Turner

Applied Research Lab
Computer Science & Engineering
Washington University

www.arl.wustl.edu

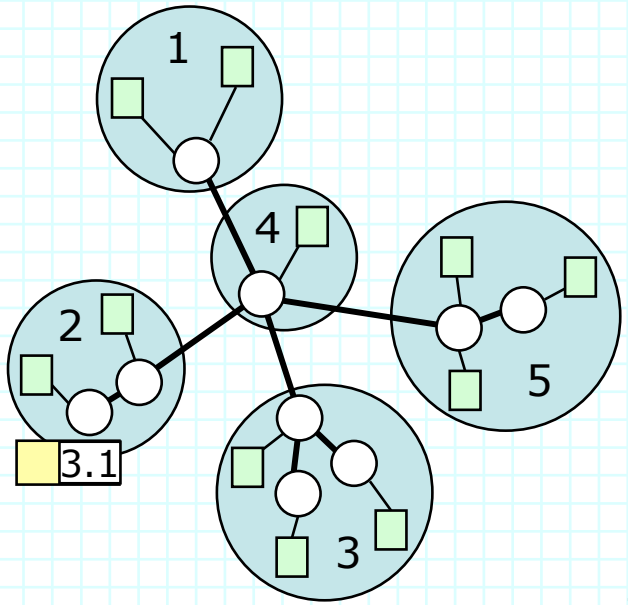
Forest Overlay Network



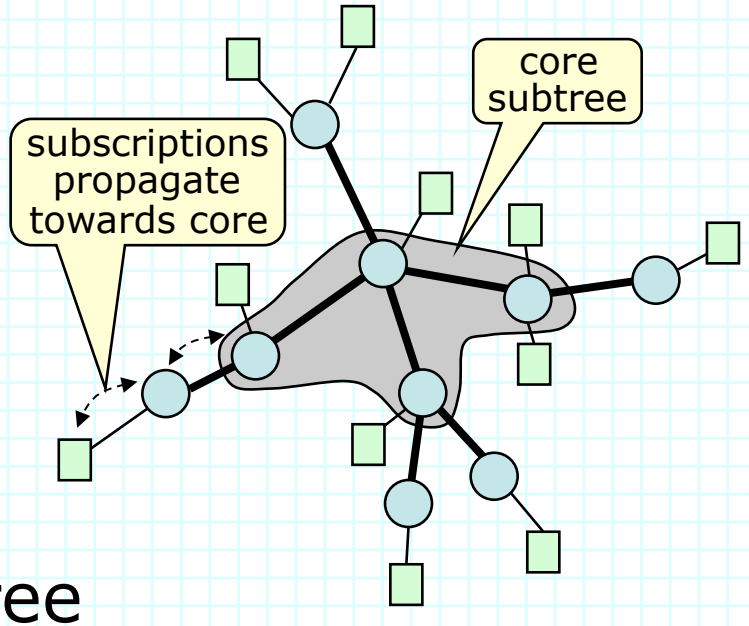
- Overlay for real-time distributed applications
  - » large online virtual worlds
  - » distributed cyber-physical systems
- Large distributed sessions
  - » endpoints issue periodic status reports
  - » and subscribe to dynamically changing sets of reports
  - » requires real-time, non-stop data delivery, even as communication pattern changes
- Per-session provisioned channels (comtrees)
  - » unicast data delivery with route learning
  - » dynamic multicast subscriptions using multicast core

Unicast Addressing and Forwarding

- Every comtree has its own topology and routes
- Two level unicast addresses
  - » zip code and endpoint number
- Nodes with same zip code form subtree
  - » all nodes in "foreign" zips reached through same branch
- Unicast routing
  - » table entry for each foreign zip code and local endpoint
  - » if no route table entry, broadcast and set route request flag
  - » first router with a route responds



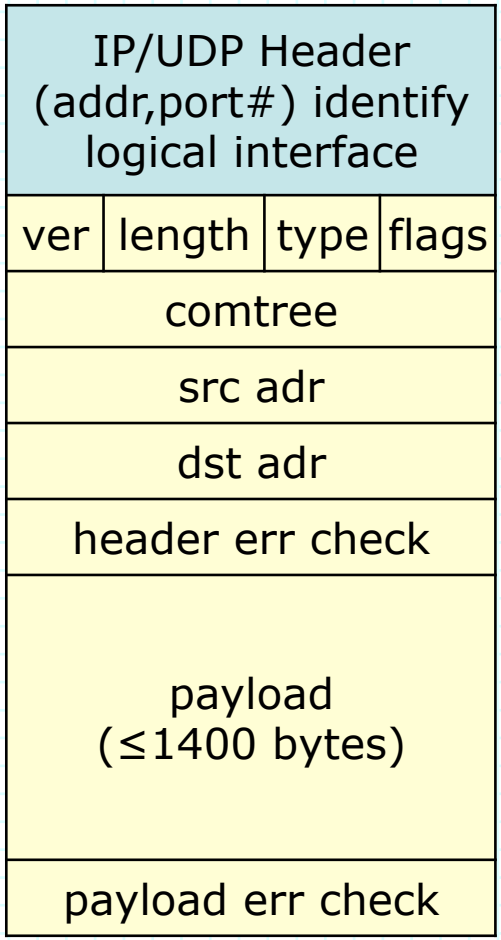
Multicast Routing



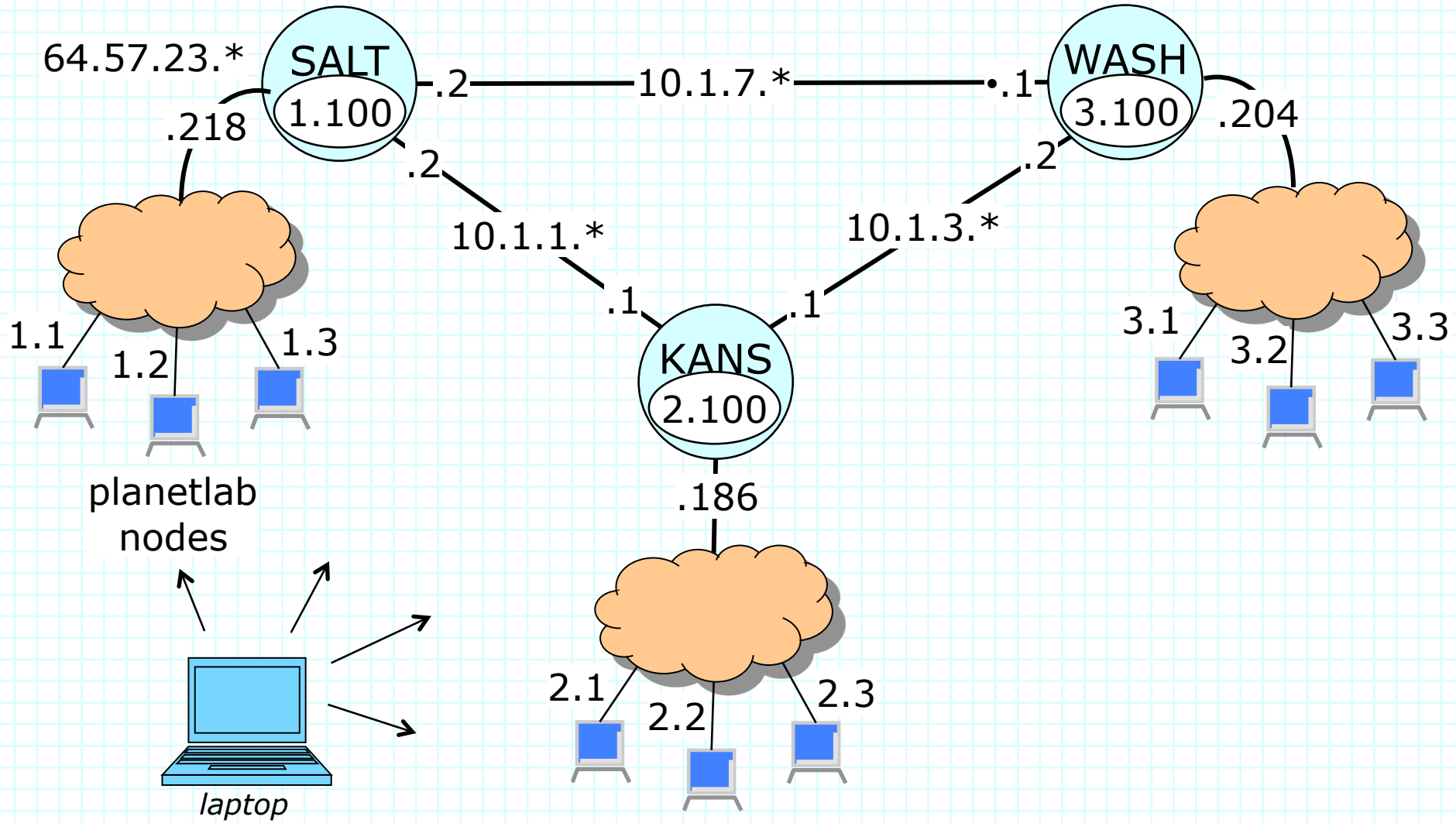
- Flat addresses
  - » on per comtree basis
- Hosts subscribe to multicast addresses to receive packets
- Each comtree has "core" subtree
  - » multicast packets go to all core routers
  - » subscriptions propagate packets outside core
- Subscription processing
  - » propagate requests towards first core router
  - » stop if intermediate router already subscribed
  - » can subscribe/unsubscribe to many multicasts at once
- Core size can be configured to suit application

Forest Packet Format

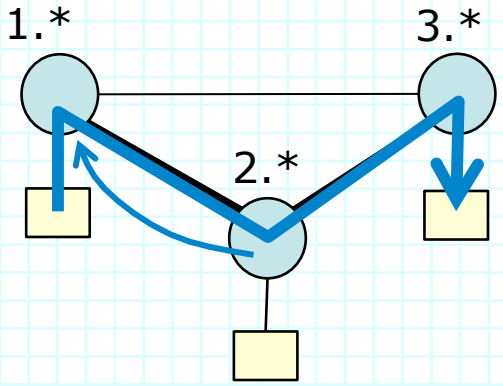
- Encapsulated in IP/UDP packet
  - » IP (addr,port#) identify logical interface to Forest router
- Types include
  - » user data packets
  - » multicast subscribe/unsubscribe
  - » route reply
- Flags include
  - » unicast route request



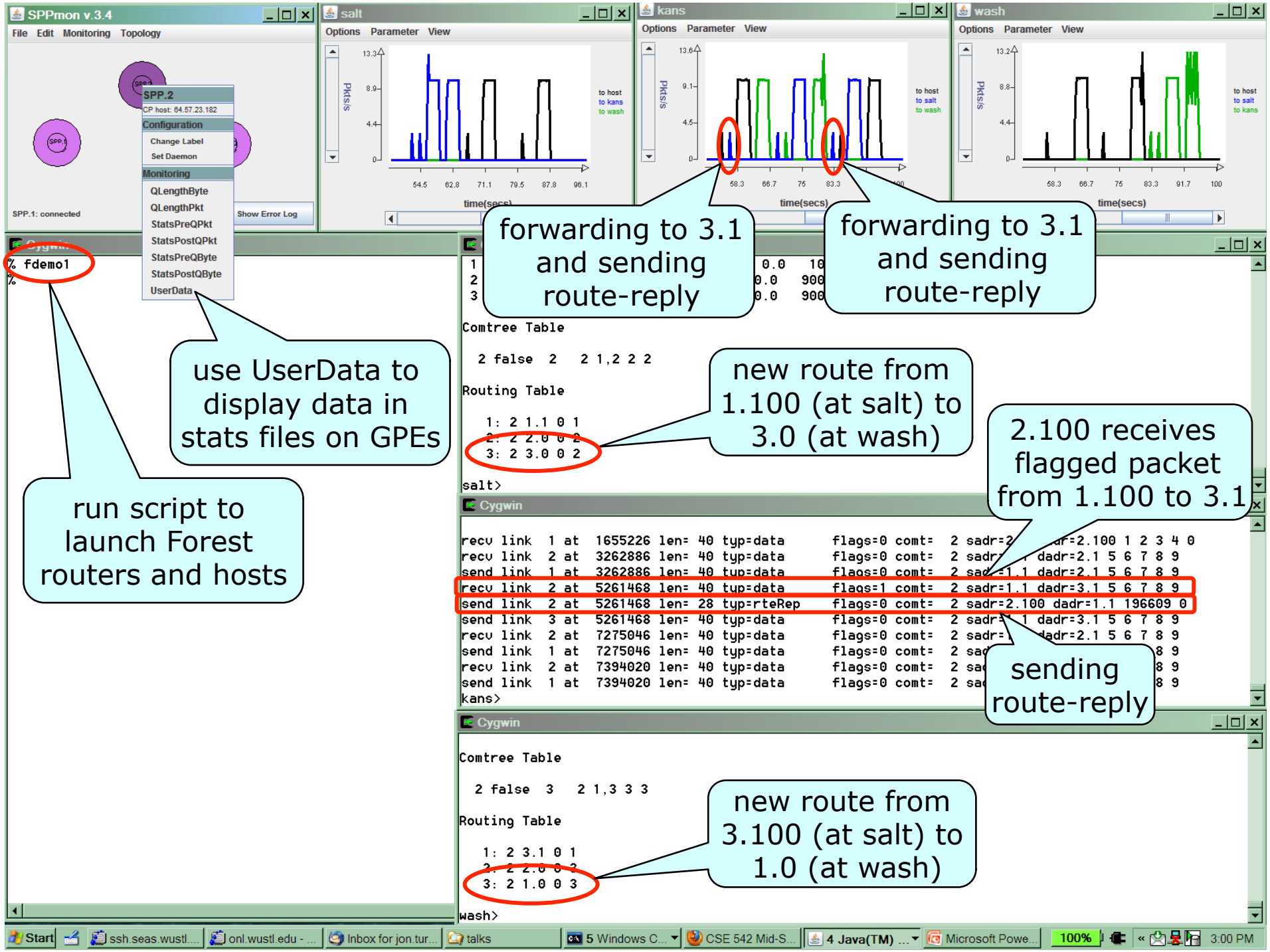
Basic Demo Setup



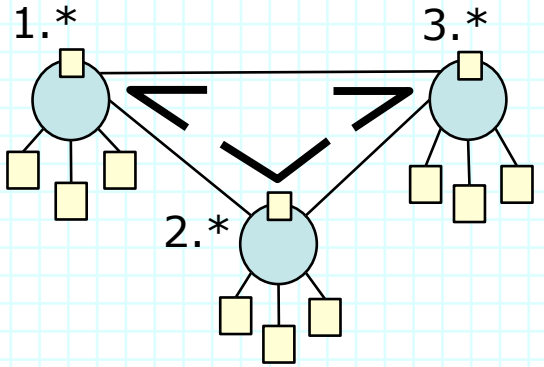
Simple Unicast Demo



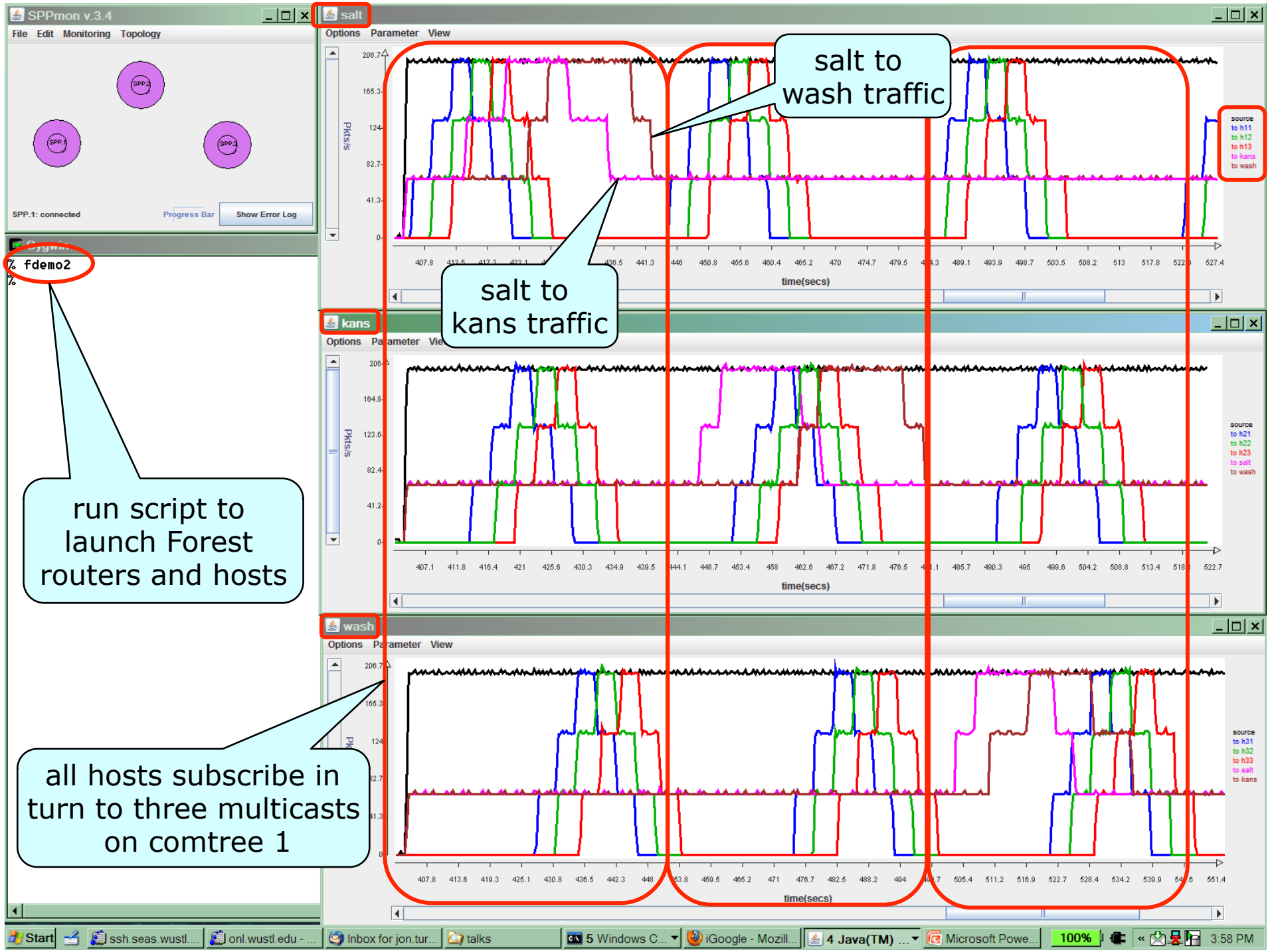
- Uses one host at each router
- Single comtree with root at KANS
- Host 1.1 sends to 2.1 and 3.1
  - » packet to 3.1 flagged with route-request and "flooded"
  - » router 2.100 responds with route-reply
- Host 2.1 sends to 1.1 and 3.1
  - » already has routes to both so no routing updates needed
- Host 3.1 sends to 1.1 and 2.1
  - » packet to 1.1 flagged with route-request and "flooded"
  - » router 2.100 responds with route-reply


SPPmon v.3.4
File Edit Monitoring Topology
SPP.2
CP host: 64.57.23.182
Configuration
Change Label
Set Daemon
Monitoring
QLengthByte
QLengthPkt
StatsPreQPkt
StatsPostQPkt
StatsPreQByte
StatsPostQByte
UserData
SPP.1: connected
Show Error Log
salt
kans
wash
Options Parameter View
Pkts/s
time(secs)
to host
to kans
to wash
to salt
forwarding to 3.1 and sending route-reply
forwarding to 3.1 and sending route-reply
% fdemo1
%
use UserData to display data in stats files on GPEs
run script to launch Forest routers and hosts
Comtree Table
2 false 2 2 1,2 2 2
Routing Table
1: 2 1.1 0 1
2: 2 2.0 0 2
3: 2 3.0 0 2
salt>
new route from 1.100 (at salt) to 3.0 (at wash)
2.100 receives flagged packet from 1.100 to 3.1
Cygwin
recv link 1 at 1655226 len= 40 typ=data flags=0 comt= 2 sadr=2 dadr=2.100 1 2 3 4 0
recv link 2 at 3262886 len= 40 typ=data flags=0 comt= 2 sadr=1 dadr=2.1 5 6 7 8 9
send link 1 at 3262886 len= 40 typ=data flags=0 comt= 2 sadr=1.1 dadr=2.1 5 6 7 8 9
recv link 2 at 5261468 len= 40 typ=data flags=1 comt= 2 sadr=1.1 dadr=3.1 5 6 7 8 9
send link 2 at 5261468 len= 28 typ=rteRep flags=0 comt= 2 sadr=2.100 dadr=1.1 196609 0
send link 3 at 5261468 len= 40 typ=data flags=0 comt= 2 sadr=1 dadr=3.1 5 6 7 8 9
recv link 2 at 7275046 len= 40 typ=data flags=0 comt= 2 sadr=1 dadr=2.1 5 6 7 8 9
send link 1 at 7275046 len= 40 typ=data flags=0 comt= 2 sad 8 9
recv link 2 at 7394020 len= 40 typ=data flags=0 comt= 2 sa 8 9
send link 1 at 7394020 len= 40 typ=data flags=0 comt= 2 sa 8 9
kans>
sending route-reply
Comtree Table
2 false 3 2 1,3 3 3
Routing Table
1: 2 3.1 0 1
2: 2 2.0 0 3
3: 2 1.0 0 3
wash>
new route from 3.100 (at salt) to 1.0 (at wash)
Start
ssh.seas.wustl...
onl.wustl.edu - ...
Inbox for jon.tur...
talks
5 Windows C...
CSE 542 Mid-S...
4 Java(TM) ...
Microsoft Powe...
100%
3:00 PM

Basic Multicast Demo



- Uses four hosts per router
  - » one source, three receivers
- Comtree centered at each forest router
  - » each source sends to a multicast group on each comtree
- Phase 1 – uses comtree 1
  - » each receiver subscribes to multicast 1, then 2 and 3; then unsubscribes – 3 second delay between changes
  - » receivers all offset from each other
- Phases 2 and 3 are similar
  - » use comtrees 2 and 3, so different topology


SPPmon v.3.4
File Edit Monitoring Topology
SPP.1: connected
Progress Bar
Show Error Log
% fdemo2
%
run script to launch Forest routers and hosts
salt
Options Parameter View
salt to wash traffic
salt to kans traffic
Pkts/s
time(secs)
source
to h11
to h12
to h13
to kans
to wash
kans
Options Parameter Vie
to h21
to h22
to h23
to salt
to wash
wash
Options Parameter View
all hosts subscribe in turn to three multicasts on comtree 1
to h31
to h32
to h33
to salt
to kans
Start
ssh.seas.wustl....
onl.wustl.edu - ...
Inbox for jon.tur...
talks
5 Windows C...
iGoogle - Mozill...
4 Java(TM) ...
Microsoft Powe...
100%
3:58 PM

Setting up & Running the Demo

- Prepare Forest router configuration files
  - » config info for Forest links, comtrees, routes, statistics
- Prepare/save SPPmon config – specify charts
- Reserve SPP resources
  - » bandwidth on four external interfaces (one for sliced)
- Start session
  - » claim reserved resources
  - » setup communication endpoint for router logical interfaces and for sliced to report statistics
  - » start sliced on SPP and SPPmon on laptop
- Start Forest routers & hosts, then observe traffic
  - » done remotely from laptop, using shell script

Reservation Script

reservation start and end times (mmddhhmm) GMT

- On spp, execute **`reservation 03151000 03152200`**

copy reservation to a file

```
#! /bin/bash
cat >res_file.xml <<foobar
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<spp>
  <rsvRecord>
    <rDate start="2010${1}00" end="2010${2}00" />
    <plRSpec>
      <ifParams>
        <ifRec bw="10000" ip="64.57.23.186" /> ...
      </ifParams>
    </plRSpec>
  </rsvRecord>
</spp>
foobar
scfg --cmd make_resrv --xfile res_file.xml
```

reserve interface bandwidth (4 of these)

invoke scfg on reservation file

Setup Script

- On spp, execute **`setup`**

```
#! /bin/bash
# claim reserved resources
scfg --cmd claim_resources
# configure interfaces, binding port numbers
scfg -cmd setup_sp_endpoint -bw 10000 -ipaddr 10.1.1.1
     --proto 17 -port 30123
scfg -cmd setup_sp_endpoint -bw 10000 -ipaddr 10.1.3.1
     --proto 17 -port 30123
scfg -cmd setup_sp_endpoint -bw 10000 -ipaddr 64.57.23.186
     --proto 17 -port 30123
scfg -cmd setup_sp_endpoint -bw 2000 -ipaddr 64.57.23.182
     --proto 6 -port 3551
# run monitoring daemon
cat </dev/null >stats
sliced -ip 64.57.23.182 &
```

interfaces to other SPPs

"public" interface

interface for traffic monitoring

Run Demo

- On laptop, run script **fdemo1**

```
#! /bin/sh
tlim=50          # time limit for hosts and routers (sec)
dir=fdemo1       # directory in which code is executed
...
# public ip addresses used by forest routers
r1ip=64.57.23.218 ...
# names and addresses of planetlab nodes used as forest hosts
h11host=planetlab6.flux.utah.edu
h11ip=155.98.35.7 ...
ssh ${sppSlice}@${salt} fRscript ${dir} 1.100 ${tlim} &
...
sleep 2
ssh ${plabSlice}@${h11host} fHscript ${dir} ${h11ip} ${r1ip}
1.1 ${rflg} ${minp} ${tlim} &
...
```