Routing Update Synchronization

- Another interesting robustness issue to consider...
- Even apparently independent processes can eventually synchronize
  - e.g. periodic routing protocol messages from different routers
  - thus, intuitive assumption that independent streams will not synchronize is not always valid

Synchronization

- Synchronization features:
  - abrupt transition from unsynchronized to synchronized system states
  - can be broken up by introducing randomization

Examples of Occurrence

- TCP congestion windows
  - cyclical behavior shared by flows through gateway
- Audio/video applications
  - periodic streams
- External events
  - periodic downloads
- Synchronized client restart
  - e.g. after a catastrophic failure
- Periodic routing messages
  - manifests itself as periodic packet loss on pings
How can routing messages synchronize?

- Weak Coupling when A's behavior is triggered off of B's message arrival.
- No Coupling when A sends at a time that is independent of B's message arrival.

Routing source of synchronization

- Router resets timer after processing its own and incoming updates.
- Creates weak coupling among routers.
- There are solutions:
  - Set timer based on clock event that is not a function of processing other routers' updates, or
  - Add randomization, or reset timer before processing update.

Periodic message model

- Router prepares and sends update, resets timer $T_p$ seconds after start time; received by other routers $T_r$ seconds from start.
- If router receives incoming routing update while preparing its own, router processes incoming update, $T_p$ seconds to process.
- After generating update set timer drawn uniformly from $[T_p-T_r, T_p+T_r]$ seconds, $T_p$ is avg period, $T_r$ random component. When timer expires repeat. If update occurs reflecting topology event, repeat also.

Analyzing Synchronization

- Three step approach:
  - Design a model that captures the essential behavior
  - Study the parameter space under a simulation
  - Simplify the model to make it analytically tractable.
The Periodic Message Model

A's routing update

Others' routing updates

$[T_p - T_r, T_p + T_r]$  

$T_c$

$T_d$

Triggers updates cause sending of a message before timer expires

What Happens?

Important Results

- With increasing $T_r$ (randomization)
  - takes longer to synchronize
  - may need $T_r$ to be ten times $T_c$
- A robust choice of timer $T_r = T_p/2$
- With increasing randomization, abrupt transition
  - from predominantly synchronized to predominantly unsynchronized
Routing Stability in Congested Networks

- Routing protocols exchange various "control message" for disseminating reachability information and liveness of peering sessions
- Investigate effects of control messages on stability of routing protocols
  - Focus on packet losses due to network congestion
  - Experimentation and modeling used to gain insight on protocol dynamics

Network configuration

- Study 2-node and 3-node configurations
- Link HR1 -> HR2 consistently overloaded
- Packets are dropped with $p = \frac{\lambda_f}{r}$
- Link overload $f = \frac{\lambda_r}{r}$

Methodology

- Successive routing packet losses result in peering session failure
  - Calculate two quantities
    - Mean-Time-to-Flap (U2D)
    - Mean-Time-to-Recover (D2U)
  - Use OSPF and BGP as examples of two routing protocols
    - OSPF is soft-state (periodic updates)
    - BGP works on top of TCP

OSPF Model

- HR1 sends a "hello" packet every $T_H$ (=10sec) interval
- HR2 declares HR1 down if it doesn’t receive a hello in $T_{RD}$ (=40 sec)
- $E[U2D] = \text{expected time for 4 consecutive hello packets dropped}$
OSPF model U2D time

- $E[U2D]$ = expected time to move from $S_0$ to $S_4$
- Special case for $S_4$ state due to jitter

OSPF U2D Results

- $E[U2D]$ for OSPF
  
  $\frac{20}{p^1 + p^2} + \frac{20}{p^1 + p^2} + \frac{10p + 20}{p^1 + p^2} + \frac{10}{1 + p}$

BGP Model: U2D time

- BGP uses TCP as underlying transport protocol
- Need to model successful transmission of a single BGP keepalive
  
  - TCP enforces in-order packet delivery
  - Behavior depends on TCP retransmission and RTT estimations

BGP Model: U2D time (cont.)

- $E[U2D]$ for BGP with RTT of 1 sec and HoldTime of 180 sec
  
  $\frac{1}{p^1} (1 + 2p^1 + 4p^2 + 8p^3 + 16p^4 + 32p^5 + 64p^6 + 52p^7)$
BGP model: U2D time (cont.)

- Effect of RTT on $E[U2D]$ of BGP

- As RTT increases, less retransmit opportunities for "keepalive"