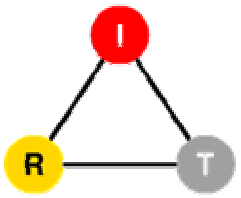
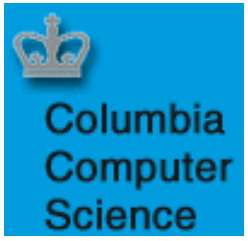
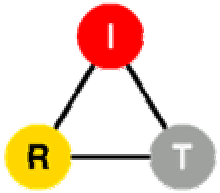


Presence

Vishal Kumar Singh and Henning Schulzrinne

April 10, 2006

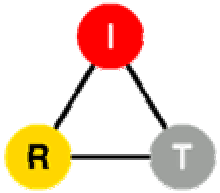




Outline

- Presence system overview
- Presence data processing
  - Presence data formats
  - Composition
  - Privacy filtering
  - Watcher filtering
  - Partial notification
- Presence security
  - Identity, authentication and authorization
  - Privacy
  - Trust
  - DoS and SPAM

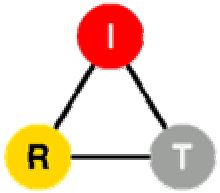
Columbia Computer Science



Outline

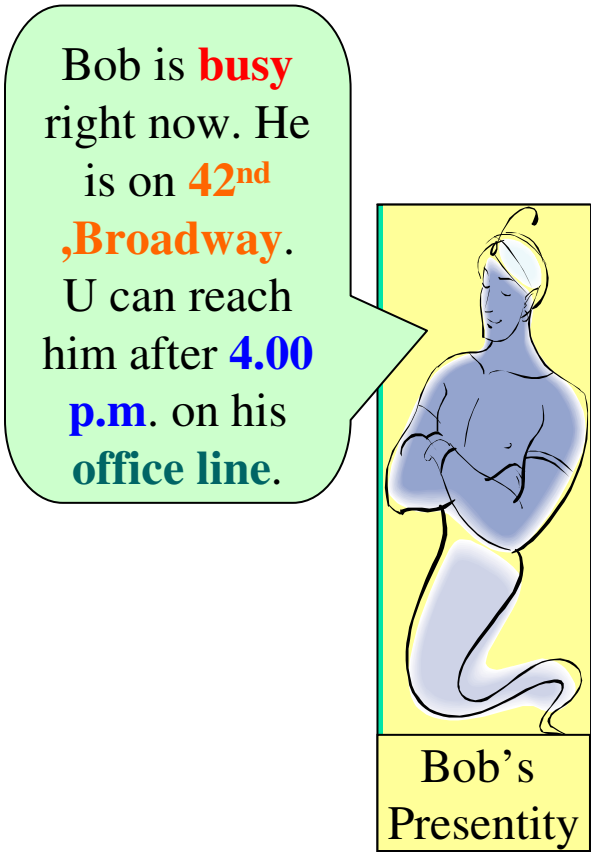
- System deployment model
  - Cross-domain deployment
  - Inter-domain deployment
  - Watcher-Info
  - XCAP
  - Event Register Package
- SIMPLEStone – Presence server benchmarking
  - Requirements
  - Issues and factors affecting presence performance
  - Architecture
  - SIMPLEStone test specification
  - Transport protocol tradeoffs

Columbia Computer Science



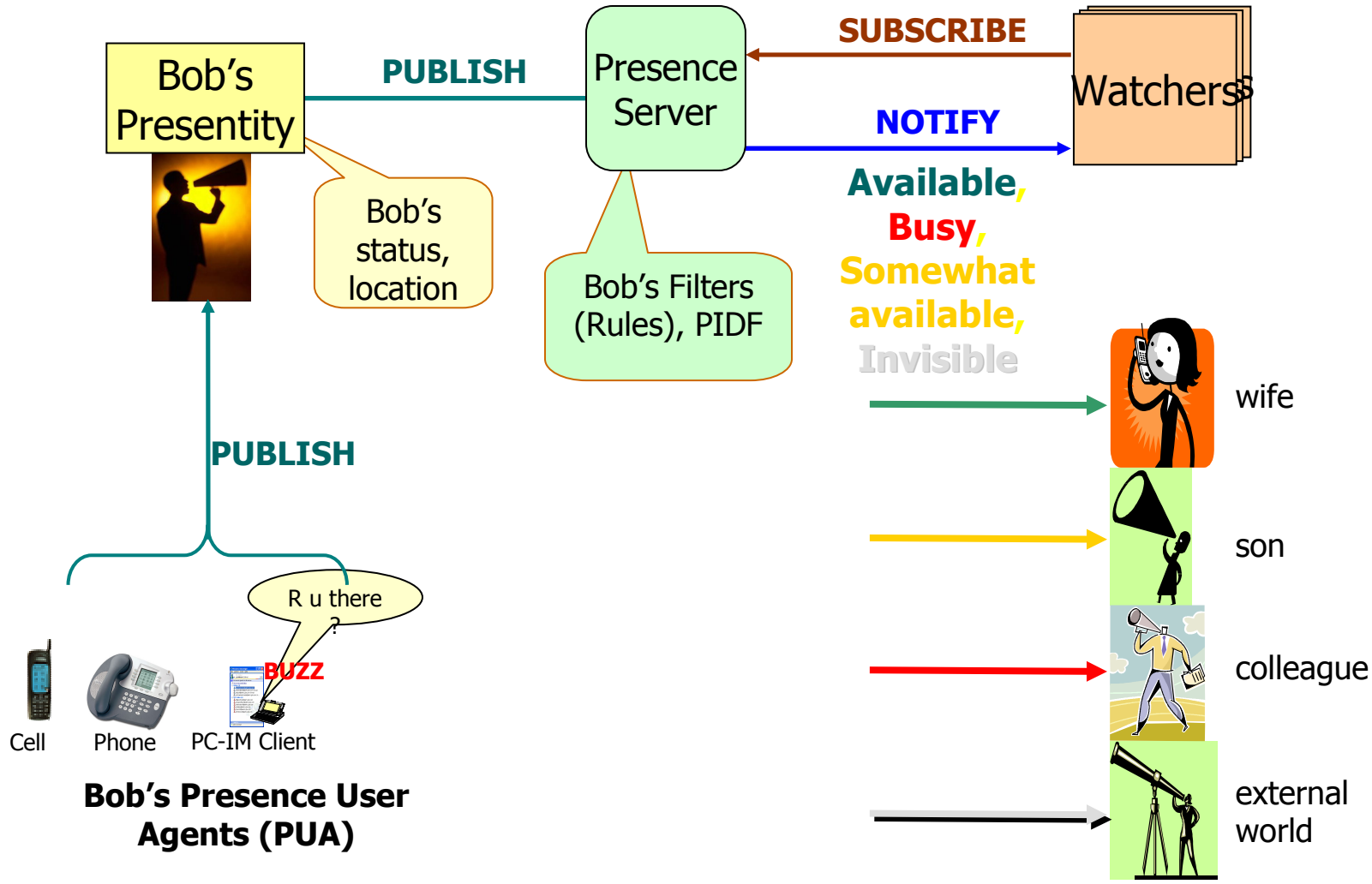
Presence System Overview

- Presence
  - Ability and willingness to communicate.
  - Rules about how and what part of presence info can be accessed
  - More detailed information includes location, preferred communication mode, current mood and activity
- Presentity
  - Represents a user or a group of users or a program
  - Source of presence information
- Watcher
  - Requester of presence information about a presentity

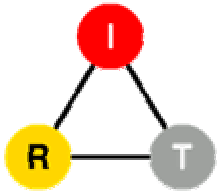


Columbia Computer Science

Presentity and Watchers



Columbia Computer Science



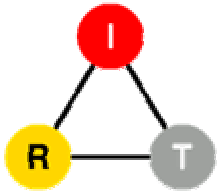
Presence System Components

- Subscription
  - Subscribe to entities
  - Authentication of subscribers
  - Subscribers specify subscription rules
- Notification
  - Updating presence state to watchers
  - Delivering presence data
  - Send notifications to the watcher in a scalable manner in real time
    - Lots of clients
    - Rate of change of data
- Publication
  - Send information to the server for distribution.
  - Multiple sources for a single address
  - Updates communications means, and capabilities

Columbia Computer Science

Presence Data Processing

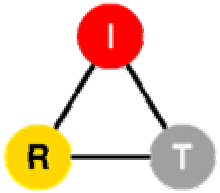
Columbia Computer Science



Agenda : Presence Processing

- Presence Data formats
  - PIDF
  - RPID
- Presence Data processing
  - Composition
    - Union composition policy
  - Privacy filtering
  - Watcher filtering
  - Partial notification

Columbia Computer Science

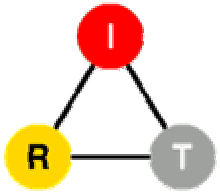


PIDF

- A presence source updates presence information by sending presence data in PIDF format
- Presentity url: specifies the "pres" url of the presentity
- Tuple "id" to uniquely identify presence information from a source
- Status: open/closed
- Optional elements
  - Communication Address: communication means and contact address of this tuple
  - Relative priority: numerical value specifying the priority of this communication address
  - Timestamp: timestamp of the change of this tuple

```
<?xml version="1.0"encoding="UTF-8"?>
<presence
xmlns="urn:ietf:params:xml:ns:pidf
"entity="sip:alice@example.com">
      <tuple id="sg89ae">
            <status>
                  <basic>open</basic>
            </status>
            <contact priority="0.8">
                sms:9845013536
            </contact>
      </tuple>
</presence>
```

Columbia Computer Science



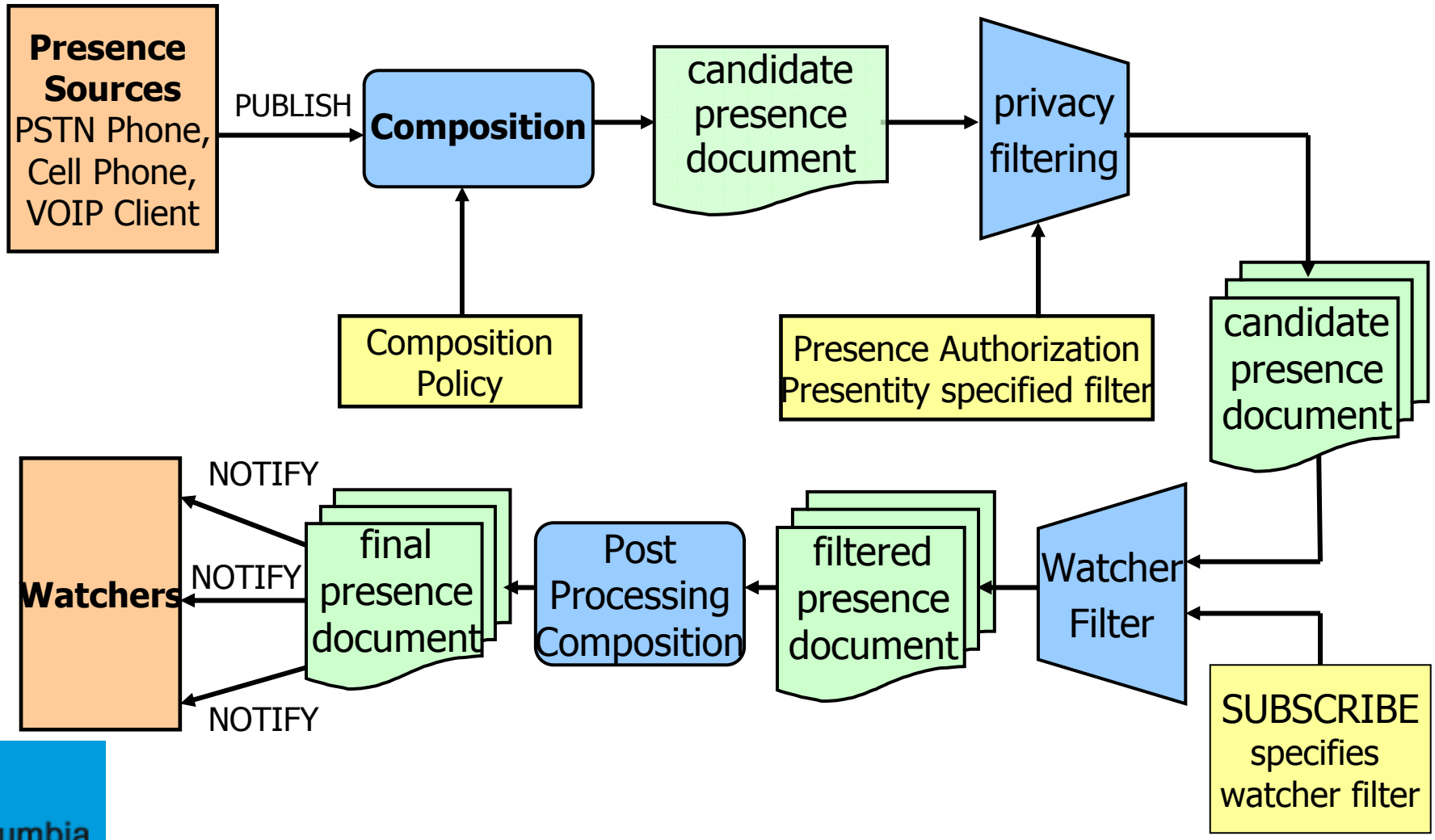
RPID

- Extension of PIDF to include detailed information about presentity like
  - What is his current activity
  - What is his mood

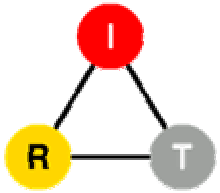
```
<presence entity="sip:alice@cs.columbia.edu">
    <tuple id="492568574609">
        <status>
            <basic>open</basic>
            <ep:activities>
              <ep:activity>On-the-phone
              </ep:activity>
            </ep:activities>
            <ep:mood>normal</ep:mood>
            <ep:privacy>
                <audio/>
                <video/>
                <text/>
            </ep:privacy>
        </status>
</tuple>
</presence>
```

Columbia Computer Science

Presence Data Processing



Columbia Computer Science

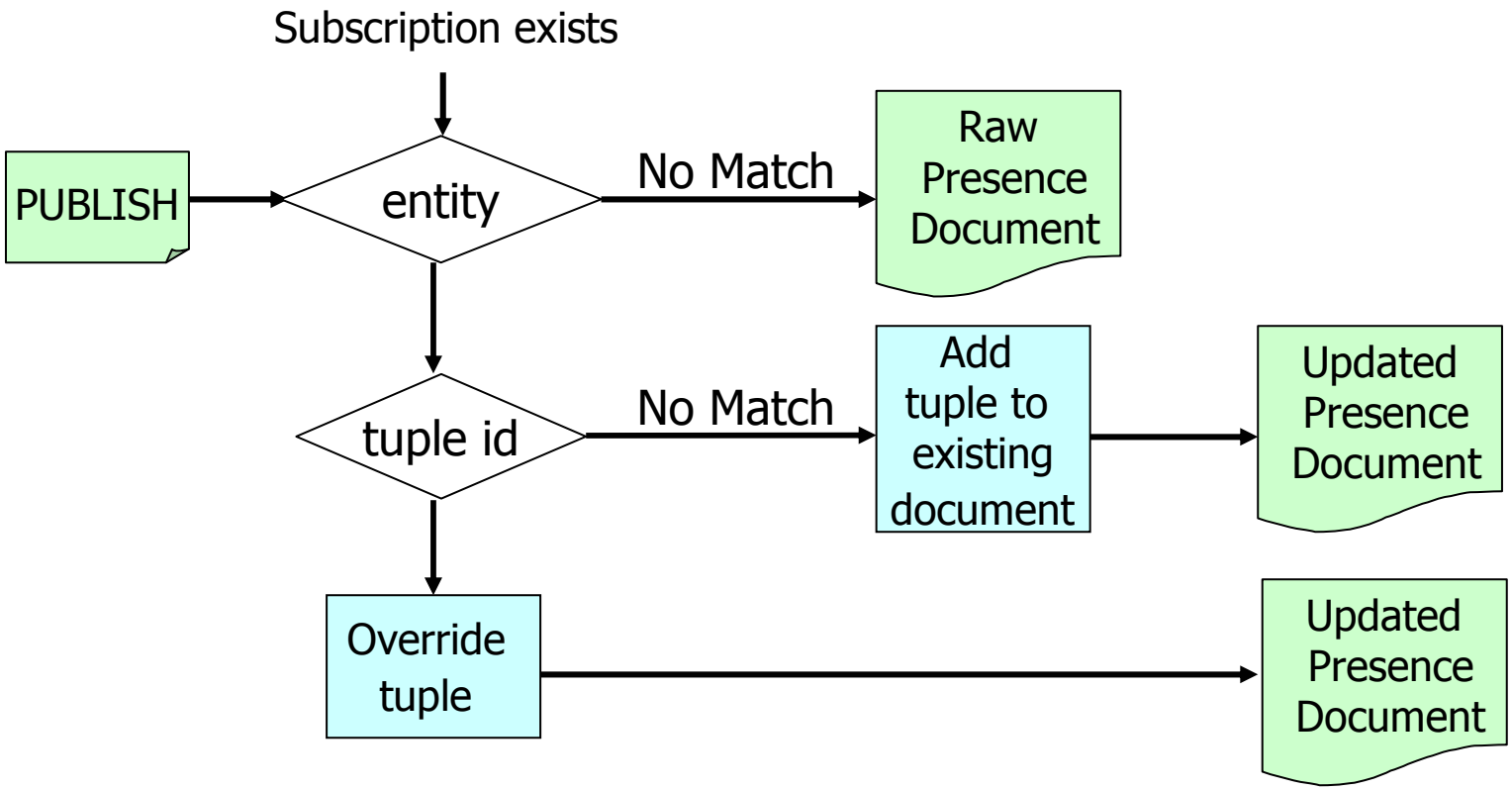


Presence Composition

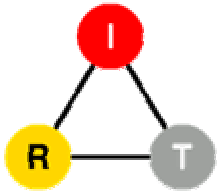
- Resolves conflicting presence information
- Sources of information conflict
  - Stale data, e.g. latest published information is old
  - Sensor failure, e.g. IM client indicates typing whereas no activity on keyboard
  - Failure to update
- Composition policies
  - Union or overriding tuples: default policy
  - Merging based on pivot elements
  - Rule based
    - Time of day type rules e.g. timestamp based
    - Source based rules e.g. source priority consideration
  - Programmatic processing of presence data using composition policy language (work going on in IETF)

Columbia Computer Science

Union Composition Policy



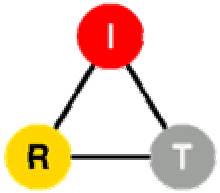
Columbia Computer Science



Composition Policy

- Based on timestamp
- Based on device priority
- Based on latest activity detected (on a device)

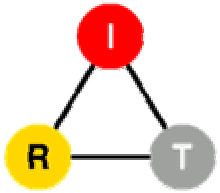
Columbia Computer Science



Privacy Filtering

- Presentity specifies rules based on which presence server
  - Allows or blocks subscriptions
  - Remove or transform presence information
- Each presentity can have zero to multiple rules
- Each rule has
  - A rule id
  - A condition – to be matched with the request
  - An action –is a transformation to be done on presence information if condition is matched.

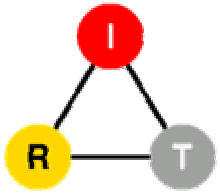
Columbia Computer Science



Privacy Filtering

- Condition
  - Attributes of requests matched against the attributes in rules
    - Identity, Sphere and Validity
- Transformation
  - A positive permission – Specifies some information which is allowed to the watcher
- The rules can be based on
  - Time of day, location, current activity etc.
  - Rules specify actions for different parts of presence information e.g. <provide-activities,

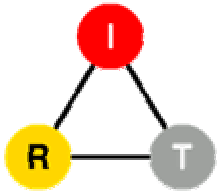
Columbia Computer Science



Presence Watcher Filtering

- Not for 'security',
  - Watchers do not want to be flooded by lots of presence information
- Allows to specify what part of presence information watcher is interested and what needs to be filtered:
  - "preferred notification information"
  - "triggers that cause that information to be delivered"
- Watchers specify filters in SUBSCRIBE requests
  - Request URI, filter URI and filter id specified in the request
- Server's local policy to honor the filter, server need not to support watcher filtering
- XPATH based filters

Columbia Computer Science



Presence Watcher Filtering

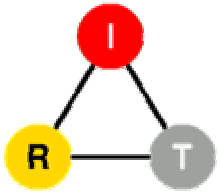
- <what> element : Results in generating NOTIFY with state information of resource only if the state of resource has changed
- <trigger> element: Results in generating NOTIFY with complete state whenever there is a change in the state of element matching the <trigger>

```
<filter
 id="123"uri="sip:presentity@example.com“>
  <what>
    <include type="xpath">
/pidf:presence/pidf:tuple[rpid:class="IM" or
rpid:class="MMS"]/pidf:status/pidf:basic
    </include>
  </what>
</filter>

<filter id="1233“
  uri="sip:presentity@example.com">
  <trigger>
    <changed from="CLOSED" to="OPEN">
/pidf:presence/pidf:tuple/pidf:status/pidf:basic
    </changed>
  </trigger>
</filter>
```

Watcher Filter format

Columbia Computer Science

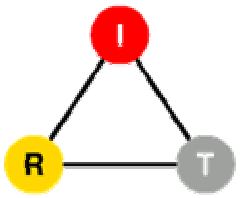


Presence Partial Notification

- For receiving only parts of the presence information that have changed since the last notification
- Watchers indicate their capability and preference to accept partial presence information using the "Accept" header.

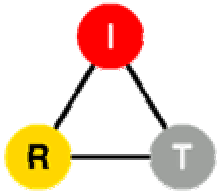
  Accept: application/pidf-partial+xml and *qalue"*
  - First NOTIFY has "state" attribute in <presence> tag in PIDF set to 'full'
  - Further NOTIFY requests have "state" as partial

Columbia Computer Science



Presence Security

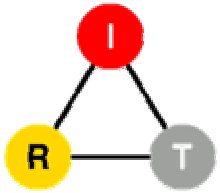
Columbia Computer Science



Agenda

- Basic Presence Security
  - Security of presence data
    - Storage and transport security
    - Confidentiality and Integrity
- Presence identity model (Authentication)
- Privacy
  - Authorization and Filtering
- Trust
  - Inter provider trust, among federations
  - Trust between users and server/service providers
- Denial of Service
  - SPAM/SPIM, attack on server
  - Attack on another UA

Columbia Computer Science

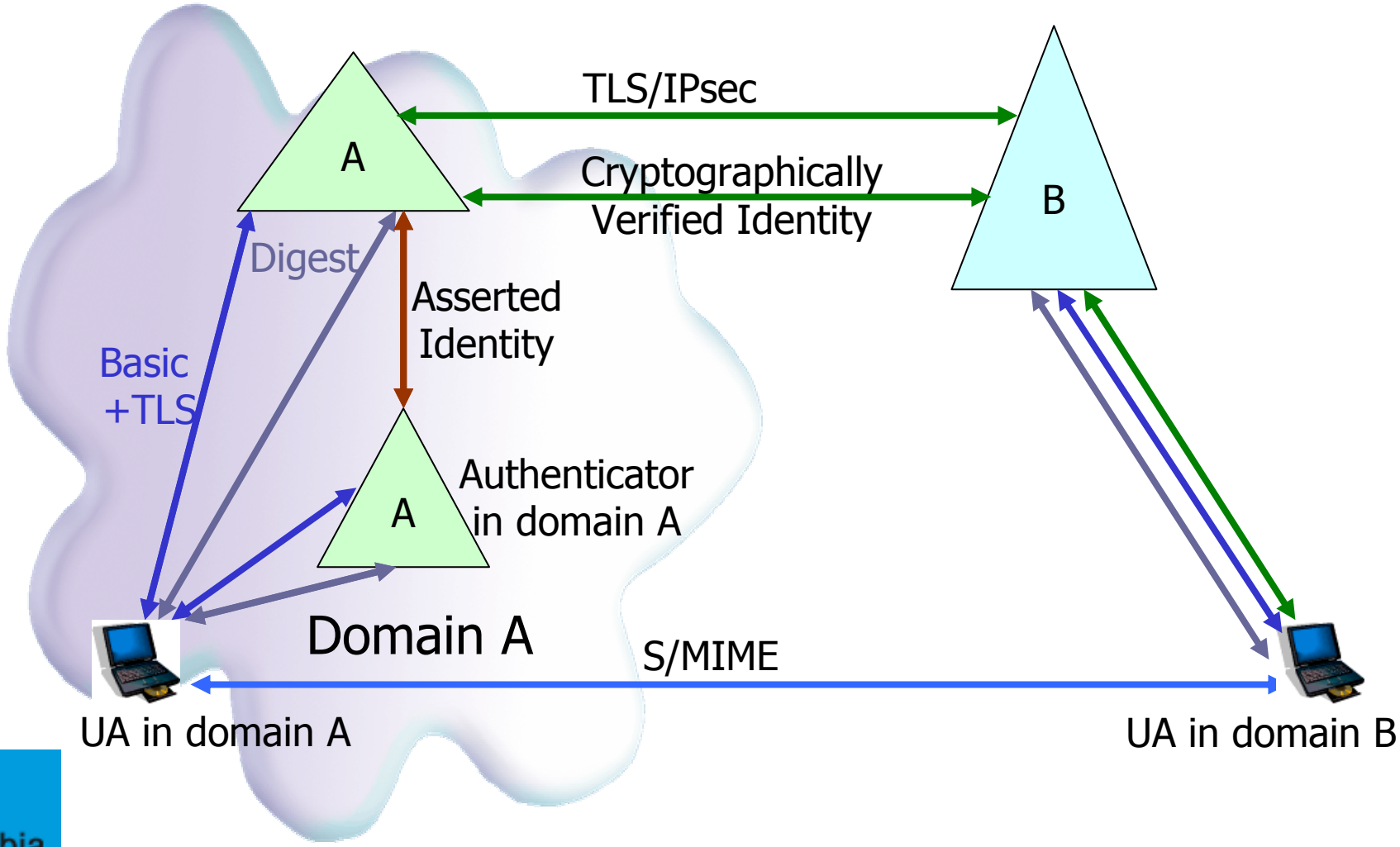


Basic Presence Security

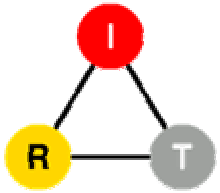
- Authentication
  - Verifying the identity of user
  - Different ways of authentication (SIP - Authentication)
  - Inter-domain Vs Intra-domain
- Authorization
  - Access and capability rights of the entity
- Encryption
  - Confidentiality, prevent MitM
  - Integrity using digests, S/MIME
  - TLS , IPSEC Between Servers , Trusted Gateways

Columbia Computer Science

Presence Authentication Scenarios



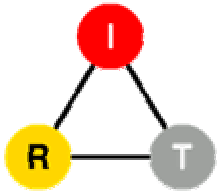
Columbia Computer Science



Presence Identity

- Presentity identification using a presence uri pres:presentity@domain
  - Identify entity using authentication (password/hash based)
  - Authentication – Are you the one you are claiming to be?
- Authorize the source to use a given identity using authentication (user credentials)
- SIP URI as identity, 1 User many identities, multiple URI's mapped to single AOR

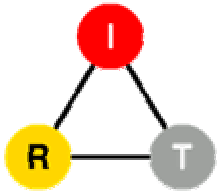
Columbia Computer Science



Presence Authorization

- Presentity specifies "block", "polite-block" or "allow" for the watchers
- Publisher authorization (currently based on authentication)
  - A source can PUBLISH on my behalf
  - A source can PUBLISH only my status, not my location.

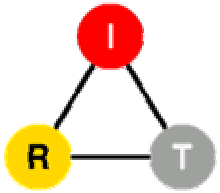
Columbia Computer Science



Privacy Filtering

- Privacy Filters "Who can see and "what"
  - Presentities specify what presence information can be given to which watchers and when
- Selective notification
  - Different views to different watchers, "my wife knows my location, not my friends"
- Providing selective access to presence information
  - <provide-devices>, <provide-services>, <provide-persons>
  - Also includes
    - <mood>, <place>, <place-type>, <relationship>, <status-icon>, <activities>
- Presence authorization policy itself is very sensitive and needs to be stored and updated securely

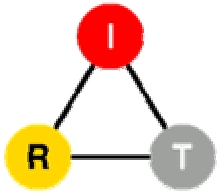
Columbia Computer Science



Privacy

- Anonymous subscription
  - Don't tell him who I am, Let me see him ?
- Hiding Identity
  - Do not reveal who I am?
- Notification confidentiality
  - Only the correct entity can see me, not a man in middle, not even the server ?
- Presence privacy to prevent use of presence information for illegal purposes, advertising, marketing, potential social harms

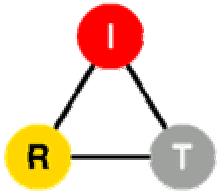
Columbia Computer Science



Denial of Service

- DoS Attack on Server
  - Authentication and Authorization of bogus requests
  - Amplification (multiple NOTIFY for genuine PUBLISH)
- DoS Attack on another watcher or UA
    - "From" header verification required (return routability)
- Presence SPAM (PSPAM)
  - Authorization + SPAM = DOS
    - Unsolicited subscriptions
  - Unsolicited NOTIFY messages
  - Use of SIP "From" header,
  - Use of SIP "Subject" line

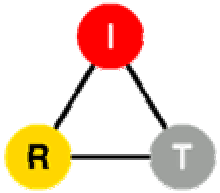
Columbia Computer Science



Presence SPAM Prevention

- Black Lists
- White Lists
- Only authorized user can send presence requests to me
- Authentication at Source
- Content filtering
- Reputation system

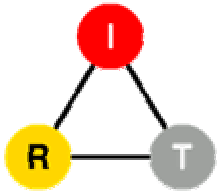
Columbia Computer Science



Trust

- Inter-Federation Trust
  - Mutual authentication : TLS, Based on certificates by a mutually trusted CA
  - Other policy negotiation among providers
- Client-Server Trust
  - Digest authentication
  - Basic authentication with TLS

Columbia Computer Science



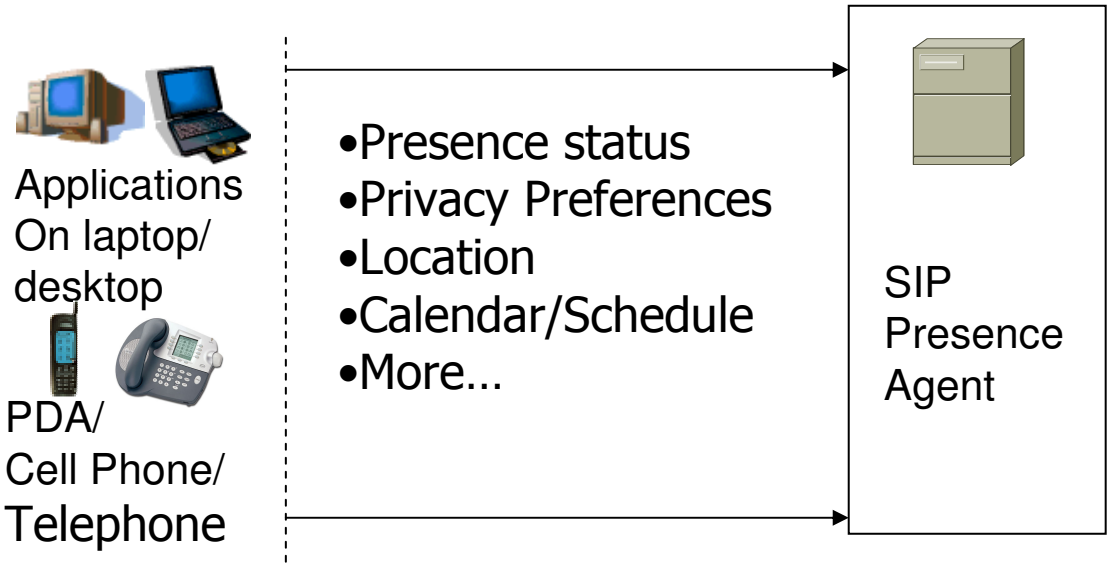
Trusting the Provider

- My presence information must not be compromised
- How much should my provider know?
- Is my availability information really private ?
- Can the Provider target advertisement on me?
- Are my privacy preferences secure? Does he know that he is polite-blocked?

Columbia Computer Science

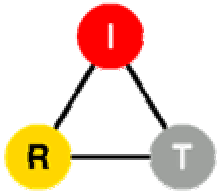
Provider Trust

- This creates a requirement : PA should be able to do data aggregation and filtering without being able to know the actual presence information



Trusting the presence service provider

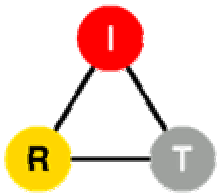
Columbia Computer Science



Presence Anonymity from Provider

- Problem Statement: Presence server must be able to perform basic composition and filtering without actually obtaining the presence information. In other words, we want to get the services like aggregation, filtering and distribution without actually revealing any presence information.
- Assumptions:
  - All watchers are authenticated not by presence server itself but by third party authenticator (authentication service). Watchers can also be authenticated by presentity itself using certificates signed by trusted CA.
  - Participating entities have public key of other entities like watchers, presentity and presence server which is originally distributed using some public key distribution mechanism.

Columbia Computer Science

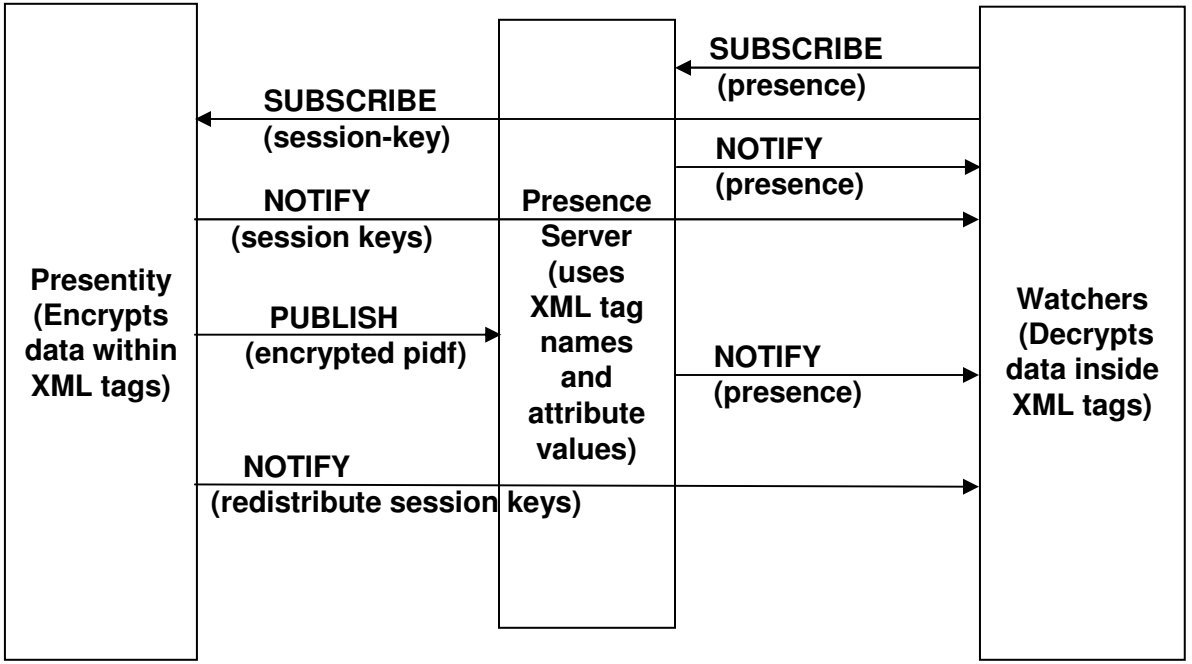


Presence Anonymity from Provider

- Proposed Solution:
  - Every presentity has 2 sets of (public, private) key pair.
  - 1st set is the public private key pair used by PKI and is installed using some PKI distribution mechanism, lets call this as (Pu1, Pr1), the second is self generated (public, private) key pair to be used only for presence purposes, lets call this as (Pu2, Pr2). The second set is used to generate a session key (SKEY) for presence.
  - When a SUBSCRIBE is received from a authenticated watcher, the presentity sends to the watcher (SKEY) encrypted using public key of watcher. Only watcher can see the (SKEY) and not the presence server or any other entity.
  - The data within the XML TAG's is encrypted using (SKEY). A slight modification to approach can be to use the second set of (public, private) key pair and distribute the public key (Pu2).
  - Higher layer XML tag names and attributes which are not encrypted are sufficient for composition and basic filtering by presence server.
  - Watchers decrypt using their private key (Pr1) and then decrypt again using the (Pu2) presence public key of presentity or the session key (SKEY) to get the presence information.

Columbia Computer Science

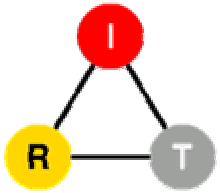
Protocol Details



Message Flow Diagram

- A direct NOTIFY is sent with 'session key' to the watchers
- Any change in 'session key' causes a notification containing the key to be sent to the watchers
- This can be integrated with presence architecture based certificate distribution as proposed in Certificate Management Service for SIP draft draft-ietf-sipping-certs.

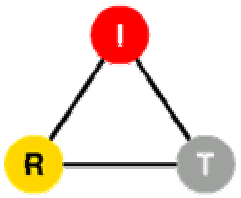
Columbia Computer Science



Presence Anonymity from Provider

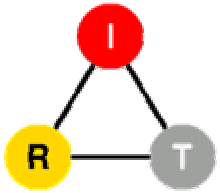
- Open issues: To be investigated further
  - Key distribution, How, to distribute the (Pu2) to all watchers securely. Probably in a direct initial NOTIFY and encrypted using watchers public key
  - Pu1 Key leakage, assume one of the watchers is compromised. One solution is presentity regenerates the keys and distributes in a notification to all current subscribers periodically and whenever an existing subscriber is blocked
    - May be Cyclic keys can solve
  - Basic composition works, But does filtering still work ?
  - The approach works if watchers are non anomalous, hence the only requirement is a strong authentication system for watchers

Columbia Computer Science



Presence Deployment

Columbia Computer Science

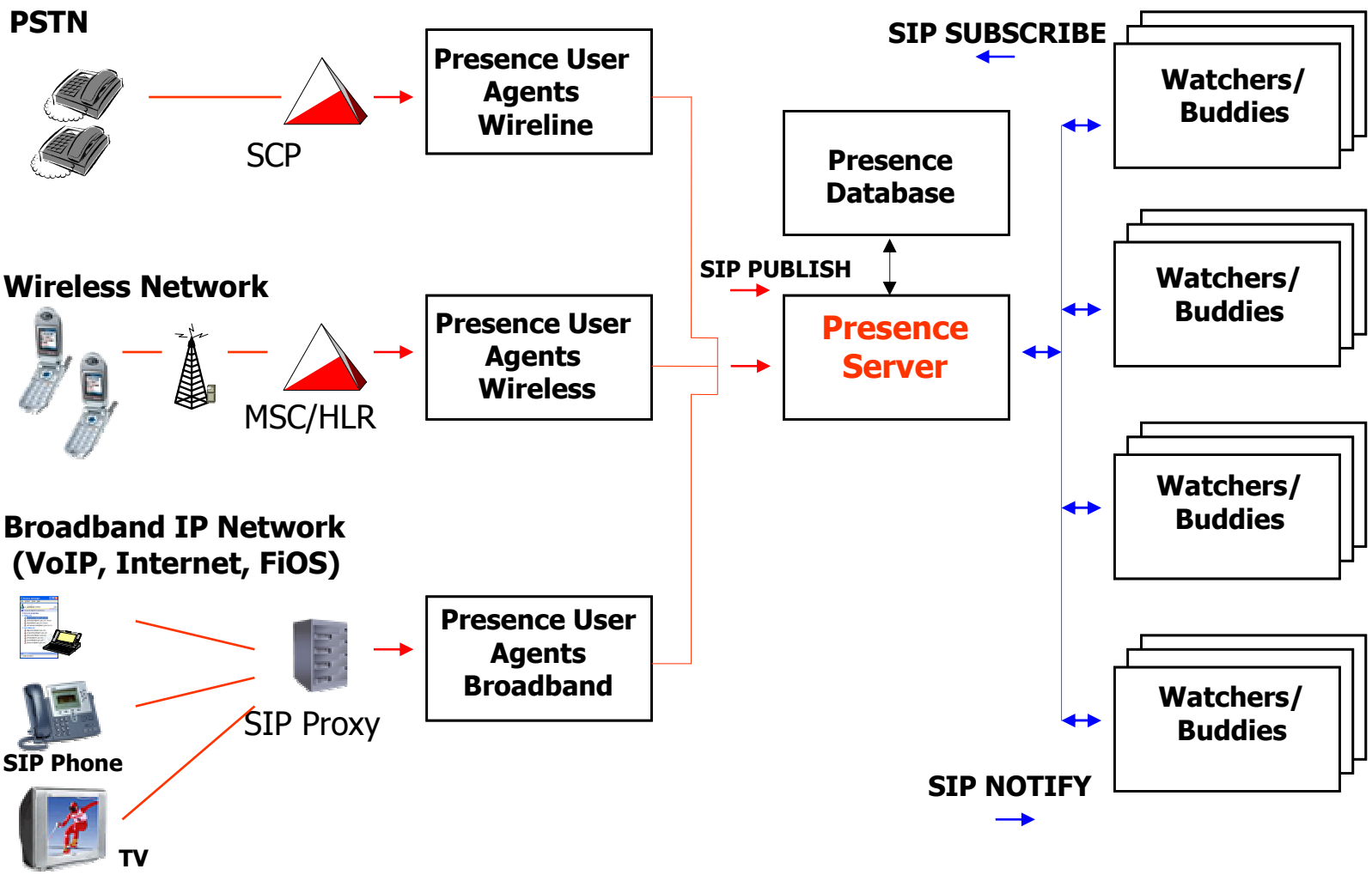


Presence System Deployment

- Cross Domain Model
  - PSTN, Cellular and VoIP worlds
- Inter Federation Model
- Different Components
  - XCAP Server
    - Buddy List
    - Presence Rules
    - Resource Lists
  - Resource List package
  - Watcher Info Package
  - Event Registration Package

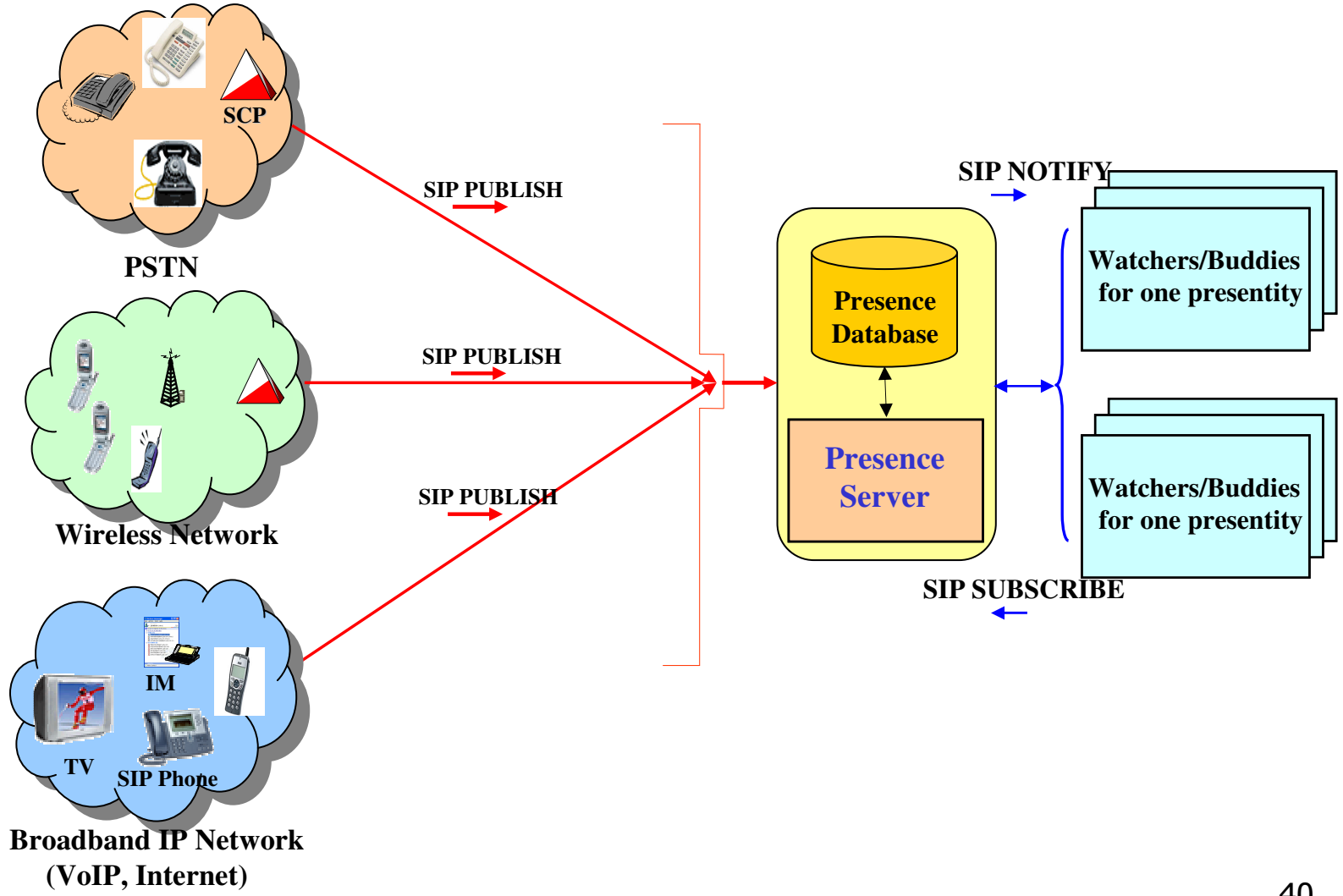
Columbia Computer Science

Cross-domain Presence Deployment



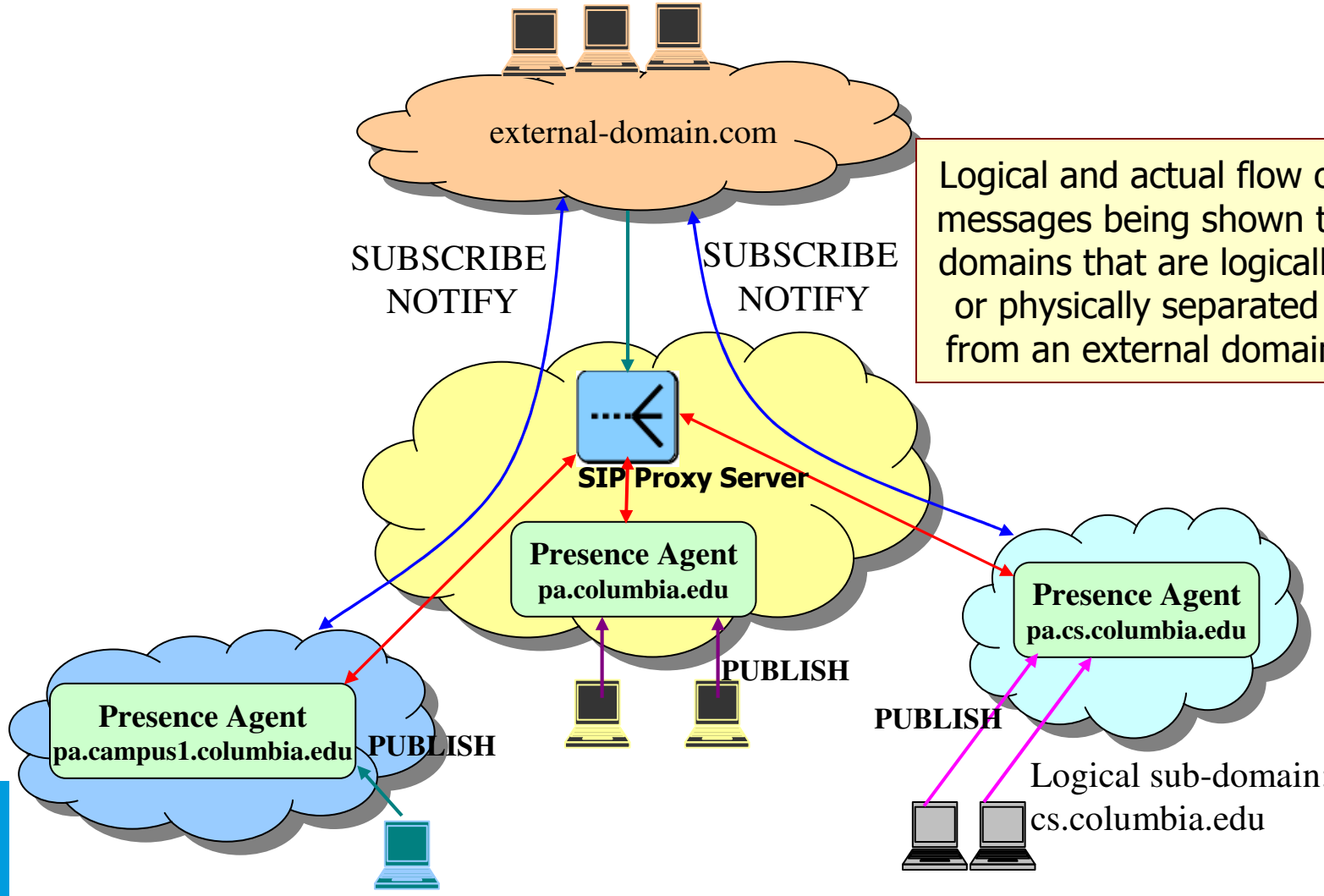
Columbia Computer Science

Cross-domain Presence Deployment



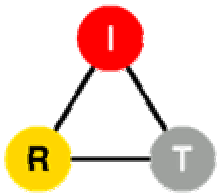
Columbia Computer Science

Inter-domain presence: Cross federation (logical and physical)

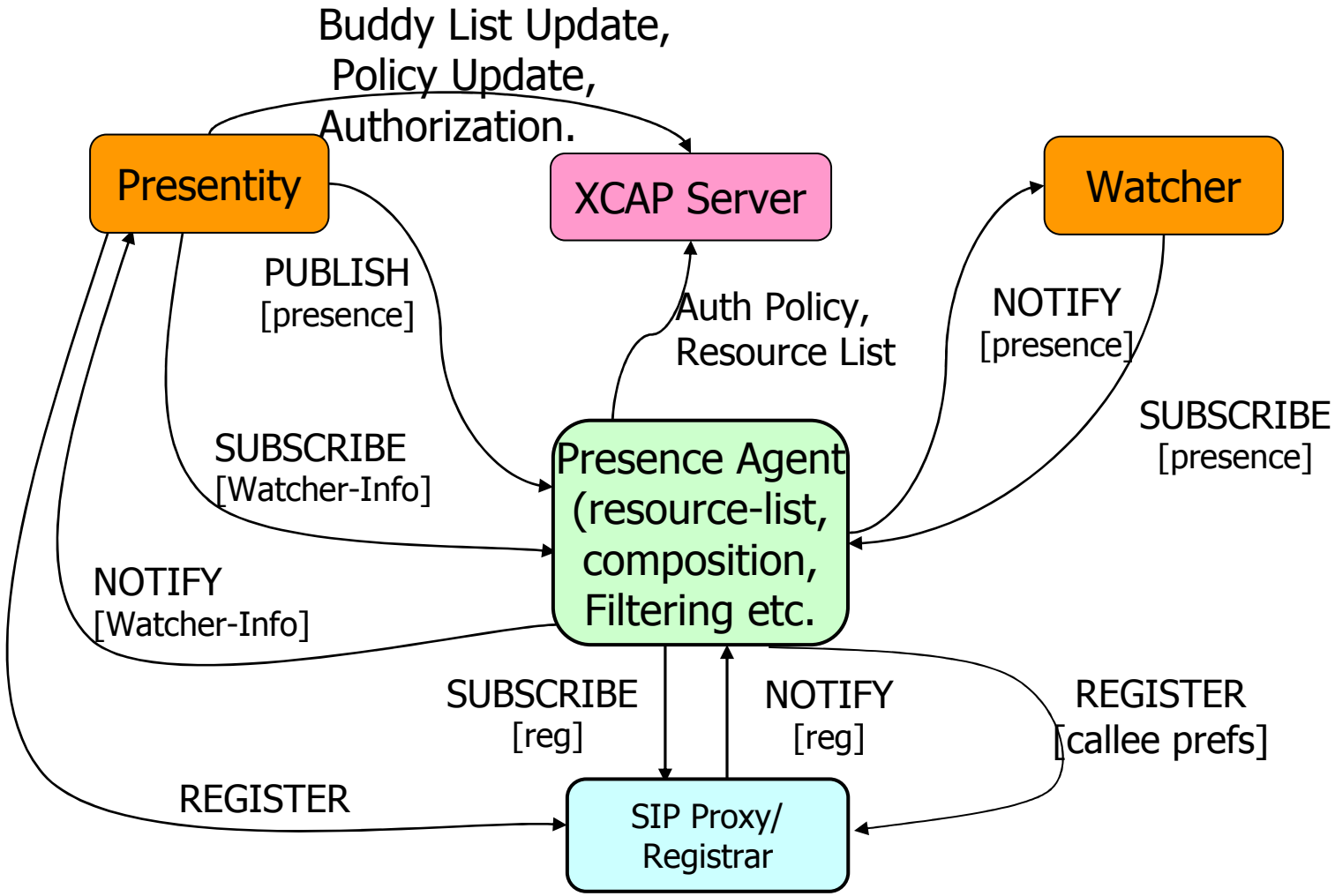


Logical and actual flow of messages being shown to domains that are logically or physically separated from an external domain

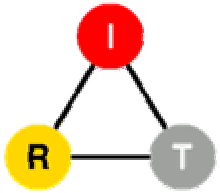
Columbia Computer Science



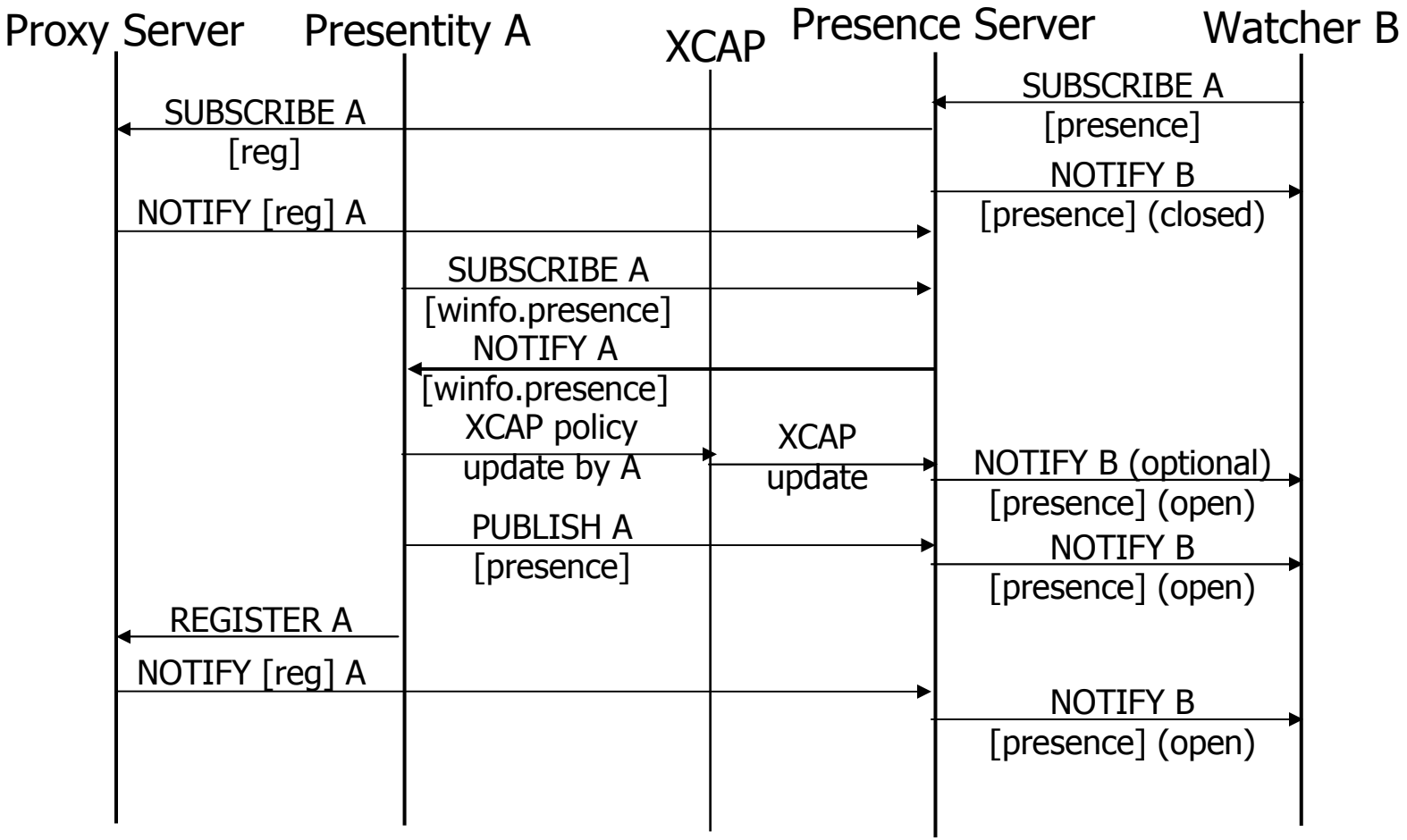
Interaction between different presence protocols



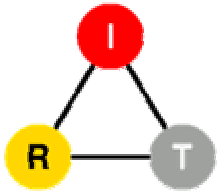
Columbia Computer Science



Presence Message Flow Diagram



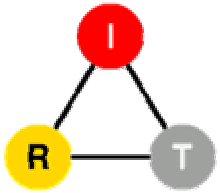
Columbia Computer Science



XCAP

- HTTP based protocol to read, write and modify XML based application configuration data
  - Creating and modifying presence authorization policy file
  - Creating and modifying the buddy list and presence resource lists
- XPATH based - XML mapped to HTTP URI
- XCAP security
  - Presentity can modify only their own data
    - Based on home directory or directory service
  - Use of HTTPS

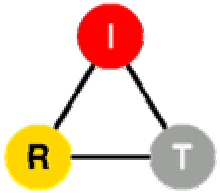
Columbia Computer Science



Watcher-Info Event Package

- Presentity needs to get the changes in the watcher information for an event package
  - Presentity subscribes to watcher-info package for a given event package
  - Presentity gets NOTIFY containing list of watcher's for that event package
  - Presence server sends notification to the presentity whenever presentity's watcher information changes, i.e.,
    - Existing watchers becoming active, pending
    - Changes in the watcher list (subscribe, unsubscribe)
  - Used by presentity to make authorization decisions and create presence privacy policy

Columbia Computer Science



Watcher-Info

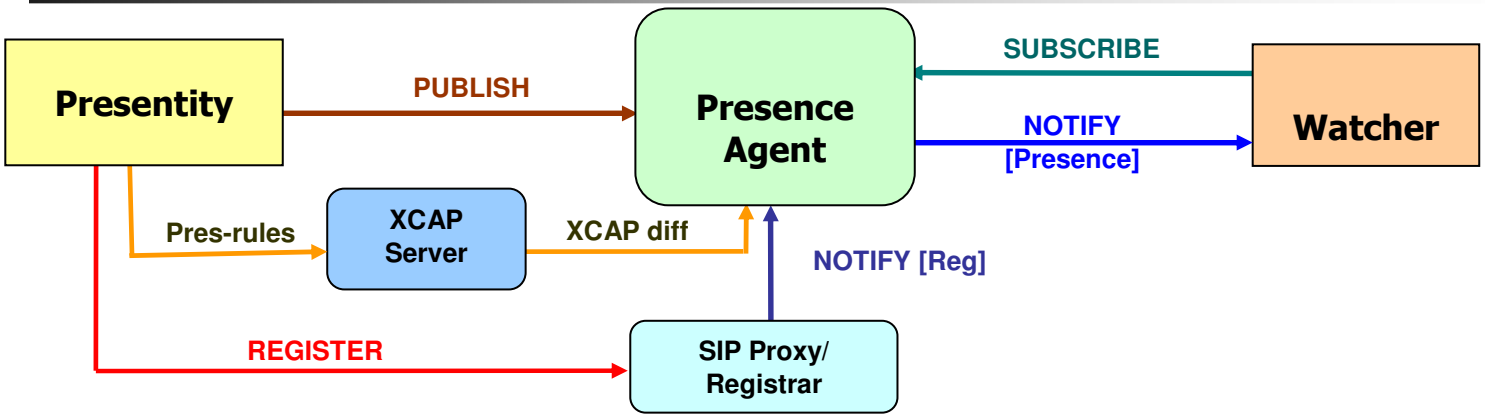
- Winfo Authorization
  - Presentity is authorized by default to get its own watcher-list
  - Presentity can authorize others to see its watchers
- Presentity can see the watchers of his watcher list

```
<watcherinfo
 xmlns="urn:ietf:params:xml:ns:watcherinfo"
    version="0" state="full">
<watcher-list resource="sip:B@example.com“
              package="presence">
      <watcher id="7768a77s"
            event="subscribe“  status="pending">
            sip:A@example.com
      </watcher>
  </watcher-list>
</watcherinfo>
```

Winfo-XML Format

Columbia Computer Science

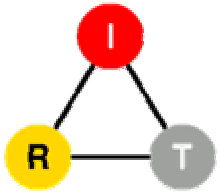
Event Registration Package



Basic block diagram of presence components

- Presence server (PA) needs to get the user's registration status from the REGISTRAR or the proxy server
  - Registration is a source of user's presence information
- PA subscribes to SIP proxy server (registrar) with event=register to get notification of changes in users registration status
- User registration state changes when
  - New Register request arrives
  - Registration time-out or expiry
- SIP proxy server sends NOTIFY to PA which updates presence state and distributes it to the watchers

Columbia Computer Science

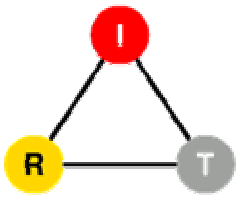


Event Registration Package

```
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo“
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    version="0“  state="full">
   <registration aor="sip:user@example.com" id="as9"
            state="active">
      <contact id="76" state="active" event="registered“
            duration-registered="7322" q="0.8">
            <uri>sip:user@pc887.example.com</uri>
      </contact>
     <contact id="77" state="terminated" event="expired“
            duration-registered="3600" q="0.5">
            <uri>sip:user@university.edu</uri>
     </contact>
   </registration>
 </reginfo>
```

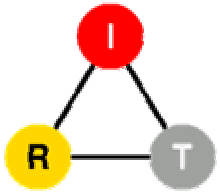
Reg-info XML format

Columbia Computer Science



SIMPLEStone – Presence Server Performance Benchmarking

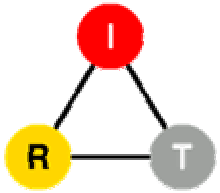
Columbia Computer Science



SIMPLEStone – Presence Server Performance Benchmarking

- Presence scalability requirements
- Need for a benchmarking metric and benchmarking tool
- Issues with presence server benchmarking
- SIMPLEStone architecture

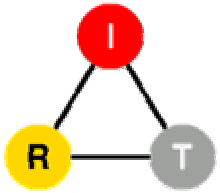
Columbia Computer Science



Presence Scalability Requirements

- To make informed, accurate decisions, presence-based services depend on the timely delivery of presence information to watchers
- Large number of watchers and presentities, with each presentity has many sources (PUA's)
- Every presentity's status change may generate a NOTIFY to all watchers.
  - Load on the network

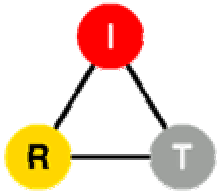
Columbia Computer Science



SIMPLEStone – Presence Benchmarking Standard

- Capacity planning and dimensioning
  - A service provider needs to know how many servers are good for a given user population
  - A server software vendor needs to specify the capacity of his server
  - Network load
- Different servers and hardware platforms
  - A uniform evaluation and performance testing methodology
  - Benchmarking server software and hardware platform performance

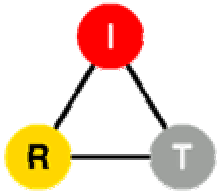
Columbia Computer Science



SIMPLEStone

- Benchmarking unit is a function of the supported user population
  - Can be expressed as the number of messages
    - rate of requests (PUBLISH, NOTIFY and SUBSCRIBE)
- Number of messages per user depends on
  - Average number of user subscriptions (buddies)
  - Notification rate to the user from buddies.
    - Device mobility
      - Cellular or wifi phone
    - User behavior
      - TV as source of presence
      - IM user has his status as the internet radio
  - Number of sources

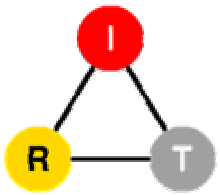
Columbia Computer Science



Issues in Presence Benchmarking

- Number of requests is not very accurate metric
- Throughput depends on
  - Protocol used (UDP, TCP or TLS)
  - Size of PUBLISH request body
  - Composition policy on server
  - Filtering support
    - Size of privacy filters
    - Size of watcher filters

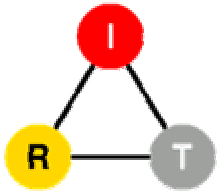
Columbia Computer Science



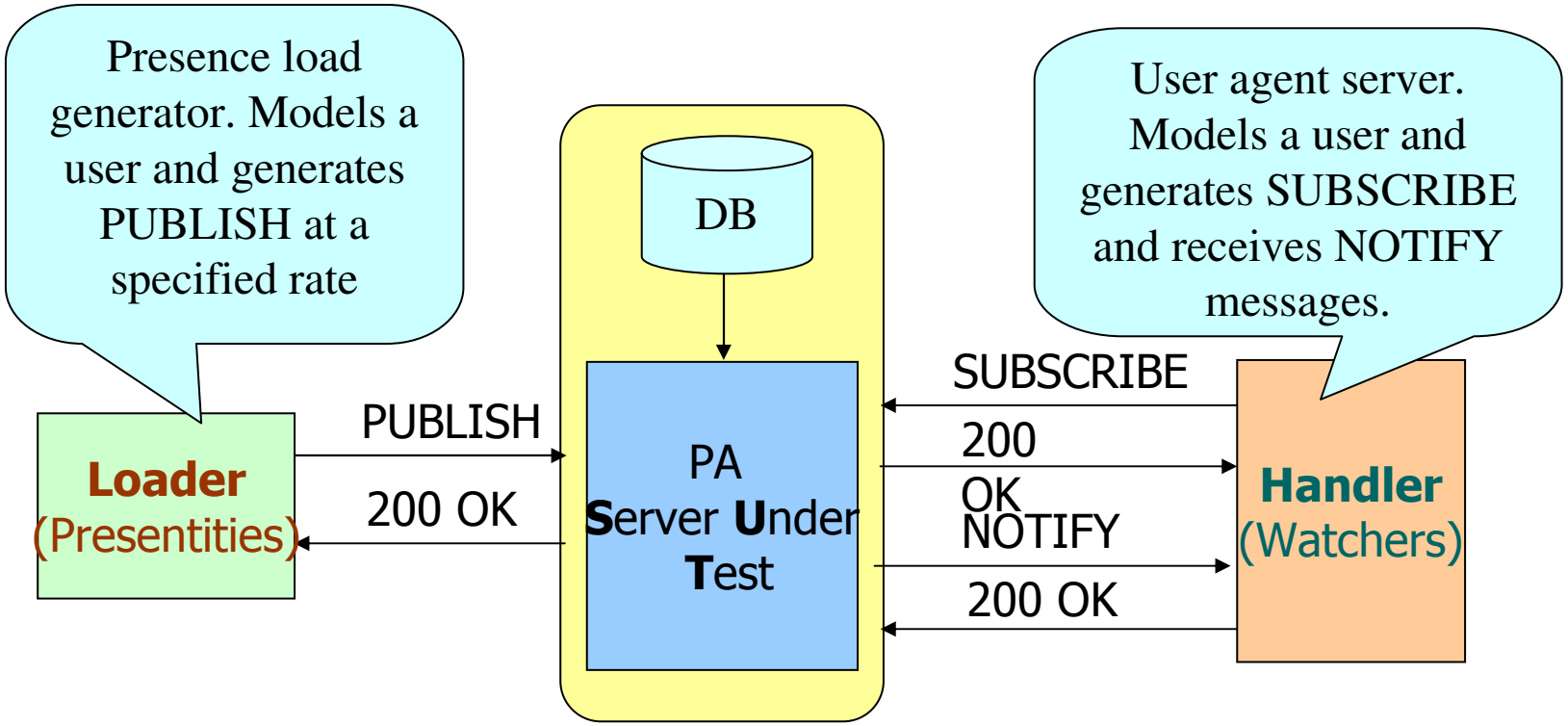
SIMPLEStone – Factors Affecting Server Performance

- Impact of composition policy
  - Single composition policy on server or per user composition.
  - Type of composition policies
    - Simple Union or Overriding
    - Intelligent Merge – Based on pivot element.
    - Rule based
      - Type of rule will effect the performance of server. Impact of Filtering
- Privacy filter and watcher filtering
  - Larger filter => more look up, comparison and XML manipulation operations on the server
  - Impact on traffic generated by the presence server
  - Rate at which watcher modifies the filter

Columbia Computer Science



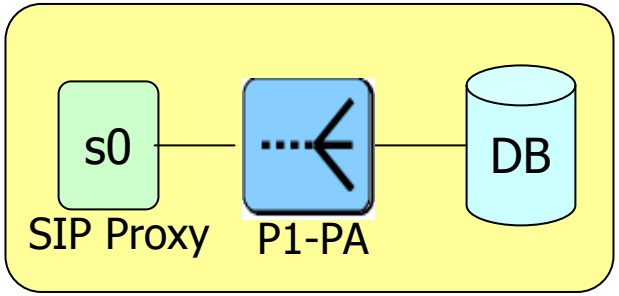
SIMPLEStone Architecture



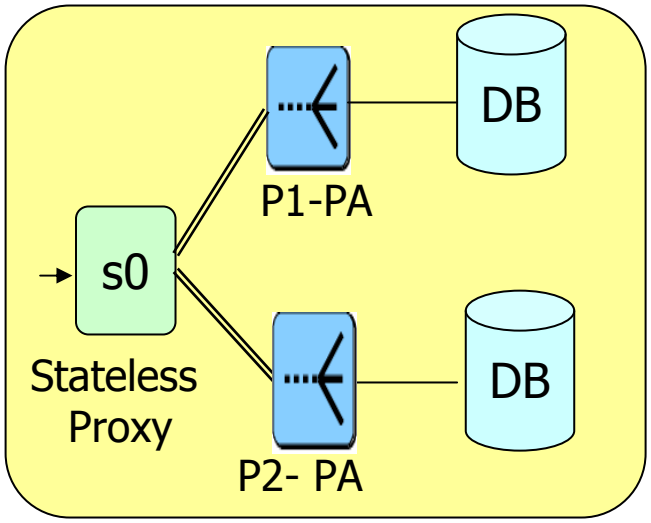
- The SUT can be replaced by different configurations in which the PA operates along with the SIP server.
- The SUT details and other test details are specified using a configuration file to the test controller.

Columbia Computer Science

SIMPLEStone SUT Configurations



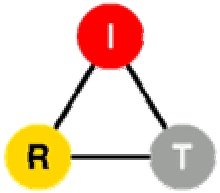
Configuration 1



Configuration 2

- SIMPLEStone sees different configurations of SUT as black box.
- The database can be arranged into 2N or N+1 redundancy mode.
- The Stateless proxy server(s) can act as load balancer distributing requests based on hashing algorithm.

Columbia Computer Science

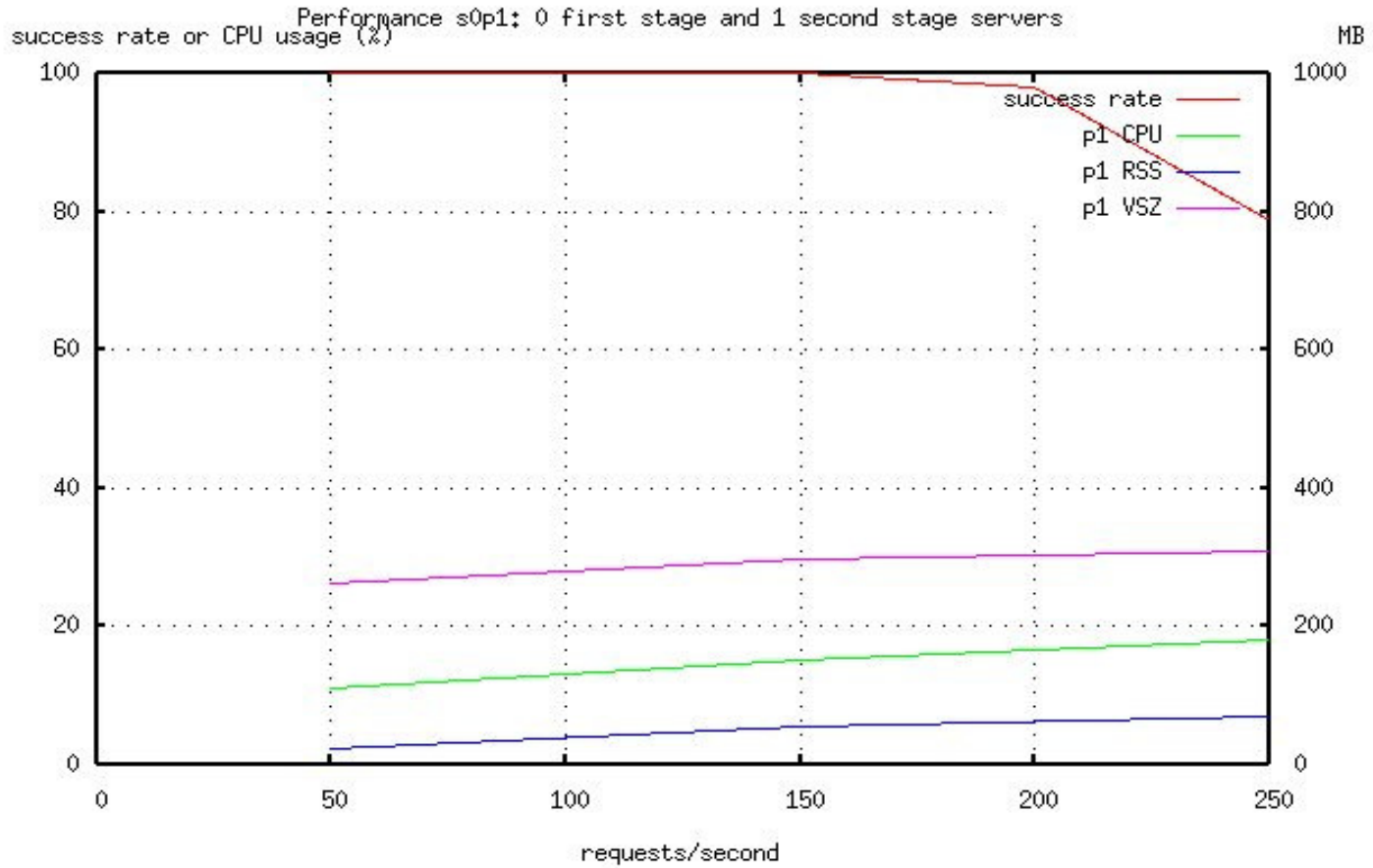


SIMPLEStone Test Details

- SIMPLEStone test specification consists of :
  - Rate at which the loader sends PUBLISH messages to the SUT
  - Number of presentities and their SIP addresses which the loader uses to generate PUBLISH and handler uses to send SUBSCRIBE
  - Number of watchers and SIP addresses for the handler's use
  - Timeout interval between receipt of NOTIFY for each PUBLISH message
  - Protocol type for the test (UDP,TCP,TLS)
  - PUBLISH message body
- Other test details
  - The machine details where loader, handler and SUT run are specified to the test controller

Columbia Computer Science

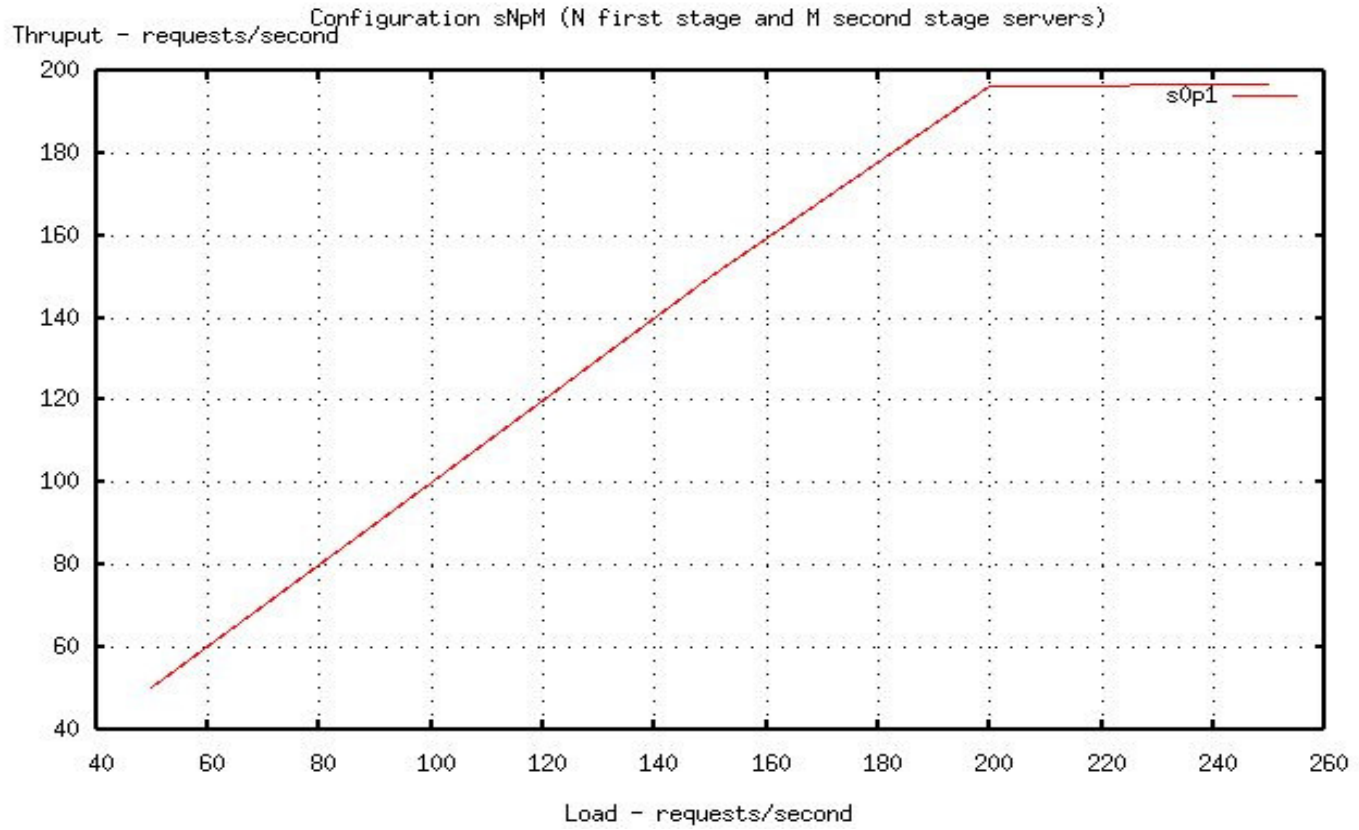
Results with SIMPLEStone tests (sipd)



success rate vs cpu and memory

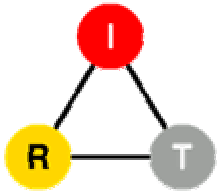
Columbia Computer Science

Results with SIMPLEStone tests (sipd)



Throughput vs Load

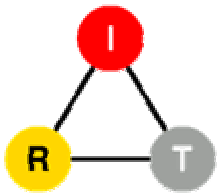
Columbia Computer Science



SIMPLEStone test analysis

- Size of SIP messages 400 -- 500 Bytes
- Size of presence message bodies (350-1000) bytes on an average, depends on source
- Best performance observed with UDP
  - 180 request per second successfully processed
- TCP: 900 open socket descriptors used up in 30 seconds for an input request rate of 30 messages per second,
- TLS: Performance goes down by 60%. Co-processor for security can be tried.

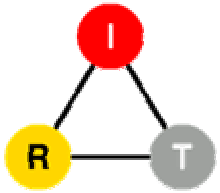
Columbia Computer Science



SIMPLEStone Test Analysis Transport Protocol

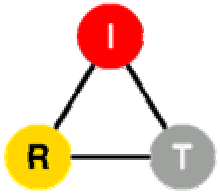
UDP	TCP or TLS
▪Low overhead, no state maintenance, Higher throughput ▪No file descriptor limit ▪No congestion control ▪NO TLS – Security. ▪Fragmentation of UDP packet is disadvantageous because of possibility of loss of fragment, Hence handling larger data sizes (NOTIFY bodies) can be an issue ▪ Client failure detection using ICMP errors, number of retransmissions depends on effectiveness of client failure detection	▪TCP state maintenance, higher overhead, lower throughput ▪File descriptor limit ▪Inbuilt congestion control ▪Security using TLS ▪ Handles larger data sizes, all fragments will have guaranteed delivery, better for large NOTIFY bodies ▪Easy failure detection during send call based on no TCP ACK. Effective failure detection to do retransmission control

Columbia Computer Science



Questions?

Columbia Computer Science



References

- RFC's (11+)
  - 3261 : Core SIP Specification
  - 2778,2779 : Presence requirements and Overview
  - 3863 : PIDF Data Format
  - 3265 : SIP Specific Event Notification
  - 3856: Presence Event Package
  - 3857, 3858 : Watcher Info Event Package and XML format
  - 3680 : Event Registration Package
  - 3840, 3841 : Indicating user agent preferences and
- Drafts (9+)
  - RPID draft
  - Common-Policy draft
  - Presence authorization rules draft
  - Watcher filtering functional description draft
  - Watcher filter format draft
  - Partial notification draft
  - Presence data model draft
  - XCAP draft
  - Resource list draft
- Reports
  - SIPStone, SIMPLEStone : Benchmarking standards
  - Presence scalability architecture,
  - Presence security report,